UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Universidad Carlos III de Madrid

GRADO EN INGENIERÍA INFORMATICA

Trabajo Fin de Grado

IMPLEMENTACIÓN DEL PROTOCOLO DE SEGURIDAD "YOKING-PROOF" EN DISPOSITIVOS MÓVILES

Autor: Carlos Alberca Pozo

Tutor: Sergio Pastrana Portillo

Septiembre de 2014

Agradecimientos

Son muchas las personas, los lugares y los momentos que han pasado a través de mi vida durante esta etapa en la Universidad.

Gracias a mi familia, pa, ma, bro, tíos, tías, abus, etc, que siempre han estado ahí apoyándome y preocupándose porque todo me saliera bien a pesar de todos los problemas con los que me haya podido encontrar.

Gracias a mi novia, Verónica, por siempre estar ahí dándome todo su apoyo.

Gracias a los amigos de la infancia que me han mostrado siempre su apoyo, entre ellos se encuentra uno muy especial que es diseñador gráfico llamado Javier Crespo Gallegos y que ha sido artífice del Logo de esta aplicación.

Gracias a todos los amigos que he hecho en la Universidad, en especial todos aquellos que seguimos juntos: Pablo, Alberto, Julián, Jaime, Iván, Mario, Lillo...

Gracias a ese Erasmus en Wroclaw que he tenido la suerte de disfrutar este curso 2013/2014, el cual me ha hecho formarme aún más como persona, mejorar los idiomas, conocer multitud de gente de muchísimos sitios, conocer múltiples ciudades y adquirir nuevos conocimientos de ingeniería informática. Gracias sobre todo a mi compañero de habitación Umut, de Turquía, con el cual he compartido más de medio Erasmus.

Y cómo me iba a olvidar, gracias a mi tutor del TFG, Sergio Pastrana, por dedicarme parte de su tiempo para apoyarme, ayudarme y resolverme dudas.

En fin, **MUCHÍSIMAS GRACIAS** de corazón todos aquellos han estado ahí durante estos últimos años!!!

RESUMEN

Actualmente nos encontramos en plena expansión de las nuevas tecnologías en la sociedad actual. Dispositivos móviles como como por ejemplo los teléfonos inteligentes, las tabletas o los relojes inteligentes, nos proporcionan prestaciones similares a las de un ordenador personal.

Estamos al principio de la era "Internet of Things (IoT)" o "Internet of You (IoY)" en las cuales todo tiende a estar interconectado en tiempo real, y donde actualmente hay 10 billones de dispositivos móviles conectados entre sí, esperando alcanzar la cifra de los 50 billones en 2020. Todo ello hace que sea un campo muy jugoso para los atacantes, desarrollando nuevas técnicas de ataque, y por lo tanto comprometiendo la seguridad y la privacidad de los usuarios.

El proyecto que hemos llevado a cabo, está basado en un protocolo de seguridad para los RFID, pero que lo hemos analizado, adaptado e implementado a los dispositivos móviles mediante el uso de la conexión bluetooth. La función principal del mismo es escanear que dos o más dispositivos pertenecen, en un mismo momento, a un mismo grupo, y de ahí su nombre comercial "Yoking-Proof" o más específico "Kazahaya".

Finalmente, para el desarrollo e implementación del mismo se han usado dos teléfonos inteligentes (Google Nexus One y Google Nexus 5), además de una placa llamada Arduino UNO con un dispositivo bluetooth.

Palabras Clave:

Internet of Things, IoT, Internet of You, IoY, RFID, Dispositivos Móviles, Bluetooth, Seguridad, Android, Arduino.

ABSTRACT

Nowadays we are in a full expansion of new technologies in today's society. Mobile devices, such a smartphones, tablets, or smartwatches give us a similar performance like a personal computer.

We are at the beginning of "Internet of Things (IoT)" or "Internet of You (IoY)" where everything tends to be interconnected in a real time. Today there are 10 billions of mobile devices connected between them, and we expect reach the figure of 50 billion in 2020. All these things together make a very attractive field for attackers, developing new attacks techniques, and thus jeopardizing user's security and privacy.

The project we have done is based on a security protocol for RFID, but we have analyzed, adapted and implemented for mobile devices by means of Bluetooth. The main function of it is to scan if two or more devices belong in the same time to the same group, so that's why we have the name "Yoking-Proof Protocol" or more specifically "Kazahaya".

In the end, for the development and implementation of it we have used two smartphones (Google Nexus One and Google Nexus 5), as well as a board called Arduino UNO with a Bluetooth device.

Palabras Clave:

Internet of Things, IoT, Internet of You, IoY, RFID, Mobile devices, Bluetooth, Security, Android, Arduino.

Contenido

Índice de tablas

Índice de figuras

CAPÍTULO 1

INTRODUCCIÓN

1.1 INTRODUCCIÓN

En estos **últimos años** hemos sido testigos del **gran crecimiento** que ha habido en las diversas **áreas de la tecnología**, llegando hasta tal punto, que es una necesidad que todo esté **interconectado** y relacionado de algún modo. Pero realmente, las primeras comunicaciones electrónicas fueron realizadas mediante telégrafos en 1800, y desde aquellas comunicaciones hasta la actualidad, han sucedido gran cantidad de cambios, como nacer el **Internet** (años 60), primera red celular (Chicago, 1977) o los actuales **Smartphones**, donde, hasta hoy, hemos conseguido tener la capacidad de almacenamiento, comunicación y potencia de un ordenador personal de hace 10 años en la palma de la mano.

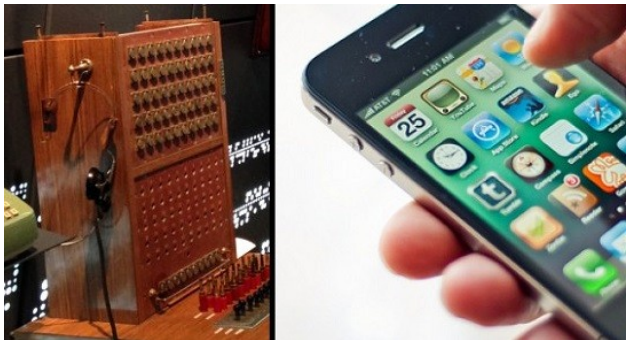


Figura 1: Desde el telégrafo al Smartphone.

Como todos sabemos, actualmente, existe una fuerte demanda de la tendencia a estar conectados entre sí, de tal modo que tengamos prácticamente **total control** sobre todo lo que nos rodea prácticamente en **tiempo real**.

En 2012, ya había **8.700 millones** de dispositivos conectados, pero durante este año (2014) se espera la verdadera explosión del famoso **"Internet of Things (IoT)"** o **"Internet of You (IoY)"**, con lo que se prevé que pasemos a unos **50.000 millones** de dispositivos en el año 2020 frente a 7.600 millones de personas en el mundo. (Cabezudo, 2014)

1.2 MOTIVACIÓN

En una sociedad donde se está incrementando el número de **dispositivos interconectados** es necesario establecer controles de **seguridad** para minimizar el impacto de potenciales intrusiones no autorizadas y evitar el uso de éstos como origen de ataques distribuidos en Internet, con el fin de proteger la **privacidad**, **intimidad** y **seguridad** de las personas. (Jaquero, 2014)

Para solventar uno de los múltiples problemas de seguridad que puede haber en los distintos dispositivos móviles, en este Trabajo de Fin de Grado se ha decidido analizar e implementar un **protocolo de seguridad** cuyo objetivo es

comprobar que dos o más dispositivos pertenecen a un mismo grupo, todo ello gracias a la tecnología **Bluetooth**.

Entre las numerosas aplicaciones de la **IoT**, está el del etiquetado digital de productos, por ejemplo en un escenario en el que se quiera controlar el stock automáticamente. En este sentido, es importante dotar de evidencias digitales que permitan por un lado controlar la presencia de los productos, y por otro realizar un análisis de en qué momento se produzco la pérdida o extravío de uno de esto. Actualmente hay varios protocolos de seguridad propuestos para este fin. Este Trabajo Fin de Grado analiza e implementa uno de estos **protocolos de seguridad** que comprueba que dos o más dispositivos pertenecen a un mismo grupo. Concretamente, la implementación hace uso de dispositivos Arduino y Smartphones con el sistema Android, usando para la comunicación la tecnología **Bluetooth**.

1.3 OBJETIVOS

El objetivo principal del TFG consiste en **analizar e implementar** un protocolo de seguridad para las **plataformas Arduino y Android**, simulando con éstas los diferentes casos de uso que puede haber en el ámbito **"Internet of Things (IoT)"**.

Para ello, vamos a enumerar una serie de objetivos que se han marcado para este proyecto:

- Estudio de la tecnología (Android, Arduino, Bluetooth) y del protocolo específico de **"Yoking-Proof"**.
- Implementación del protocolo en ambas **Android** y **Arduino**.
- Desarrollar una **aplicación móvil**, implementada en Android, que configurará los parámetros necesarios para el protocolo, además de gestionar la frecuencia de envíos **reto/respuesta**.
- Realizar un **programa** para la plataforma Arduino, el cual contendrá parte del protocolo a implementar, respondiendo así a los retos/respuesta enviados por el dispositivo Android.
- Realizar un estudio de los **tiempos y recursos** empleados en las dos implementaciones del protocolo (Android y Arduino).

1.4 ESTRUCTURA DEL DOCUMENTO

La estructura que toma este documento es la siguiente:

- **Capítulo 1:** contiene la introducción del documento, donde se le introduce al lector a una visión global del proyecto y de la documentación.
- **Capítulo 2:** contiene el Estado del Arte donde se hace una descripción de todas las tecnologías usadas tanto software como hardware, además de los contextos donde se aplican.
- **Capítulo 3:** contiene el análisis del Trabajo Fin de Grado, donde se estudian las tecnologías a aplicar.
- **Capítulo 4:** contiene en diseño detallado de cómo se han diseñado las aplicaciones y el protocolo, usando diagramas, modelos…
- **Capítulo 5:** contiene la implementación, donde se muestran alguna de las partes más relevantes de código fuente.
- **Capítulo 6:** contiene las pruebas del sistema que corroboran que todo funciona correctamente.
- **Capítulo 7:** contiene la gestión del proyecto donde se expone la planificación llevada a cabo y los costes del mismo.
- **Capítulo 8:** contiene la las conclusiones y las líneas futuras a las que hemos llegado tras la realización de este proyecto.
- **Anexos:** contiene la bibliografía usada para este proyecto, así como los manuales de usuario de las aplicaciones.

1.5 ACRÓNIMOS

IoT: Internet of Things.
IoY: Internet of You.
TFG: Trabajo Fin de Grado.
AOSP: Android Open Source Project.
QWERTY: Distribución de Teclado más común.
GPS: Sistema de Posicionamiento Global.
HTML: HyperText Markup Language.
NFC: Near Field Communication.
YAFFS: Yet Another Flash File System.
EXT4: EXtended File System for Linux.
MKV: Formato de video de Alta Definición.
TRIM: Orden para el sistema operativo en el uso del SSD.
MB: MegaBytes.
RAM: Random Access Memory
C: Lenguaje de Programación.
C++: Lenguaje de Programación.

IDE: Integrated Development Enviroment.
USB: Universal Serial Bus.
Hz: Herzios.
KB: KiloBytes
V: Voltios.
Mbps: Megabits por segundo.
dBm: decibelios relativos a un milivatio.
GHz: GigaHerzios
mm: Milímetros.
MIT: Massachusetts Institute of Technology.
IBSG: Internet Business Solutions Group.
UML: Unified Modeling Language.
OS: Sistema Operativo.
iOS: Sistema Operativo móvil de Apple.
WP: Windows Phone.
SRAM: Static Random Access Memory.
MVC: Modelo Vista Controlador.
AES: Advanced Encryption Standard.
GAP: Espacio entre algo.
APK: Application PacKage file de Android.

1.6 GLOSARIO DE TÉRMINOS

Smartphones: Teléfonos móviles inteligentes.
Arduino: Placa electrónica programable.
Android: Sistema operativo propio de Google.
Bluetooth: Tecnología inalámbrica para el área personal.
Kernel: Nucleo del Sistema operativo
Linux: Es un Sistema operativo.
Smartwatch: Relojes inteligentes.
Google: Empresa del sector de las nuevas tecnologías.
Gartner: Empresa consultora y de investigacion de las tecnologías de la información.
Dalvik: Actual Máquina Virtual de Android.
ART: Nueva Máquina Virtual de Android.
Google Maps: Programa para consultar mapas.
Tethering: Tecnología para compartir tu red de internet en el móvil.
Adobe Flash: Software para multimedia.
Google Chrome: Navegador de internet de Google.
Google Now: Asistente personal inteligente de Google.
Weareables: Conjunto de dispositivos electrónicos incorporados a prendas y complementos.
Open Source: Movimiento de software para potenciar el código abierto.
Cisco: Empresa dedicada al sector de las nuevas tecnologías.
Password: Clave que asignas a algo para protegerlo.
Debug: Depurador, programa usado para identificar y eliminar errores en los programas.

CAPÍTULO 2

ESTADO DEL ARTE

2.1 PANORÁMICA DE LAS TECNOLOGÍAS APLICADAS

En esta sección se muestra una visión global de las tecnologías estudiadas y aplicadas para la realización de este TFG.

2.1.1 Android

Alcance del sistema operativo

Android fue desarrollado por Android Inc. en el año 2003, siendo posteriormente adquirido por Google en el año 2005. La plataforma fue liberada bajo el seudónimo *"Android Open Source Project (AOSP)"* en el año 2007. Es un sistema operativo basado en el kernel de **Linux** diseñado específicamente para dispositivos móviles, como tabletas o teléfonos inteligentes (**smartphones**), y también los actuales relojes inteligentes (**smartwatch**), además de los **televisores** o **automóviles**. (Wikipedia, s.f.)

En la Figura 2 que se muestra a continuación, se observa la cuota de mercado mundial de los sistemas operativos móviles. Tal y como se aprecia en la gráfica de Gartner, se puede decir que el crecimiento de Android fue casi constante hasta 2012, pero bajando la cuota de mercado en lo que predice Gartner de 2014 a 2016, y aun así este sistema operativo seguirá con más del 50% de cuota de mercado. (Columbus, 2013)

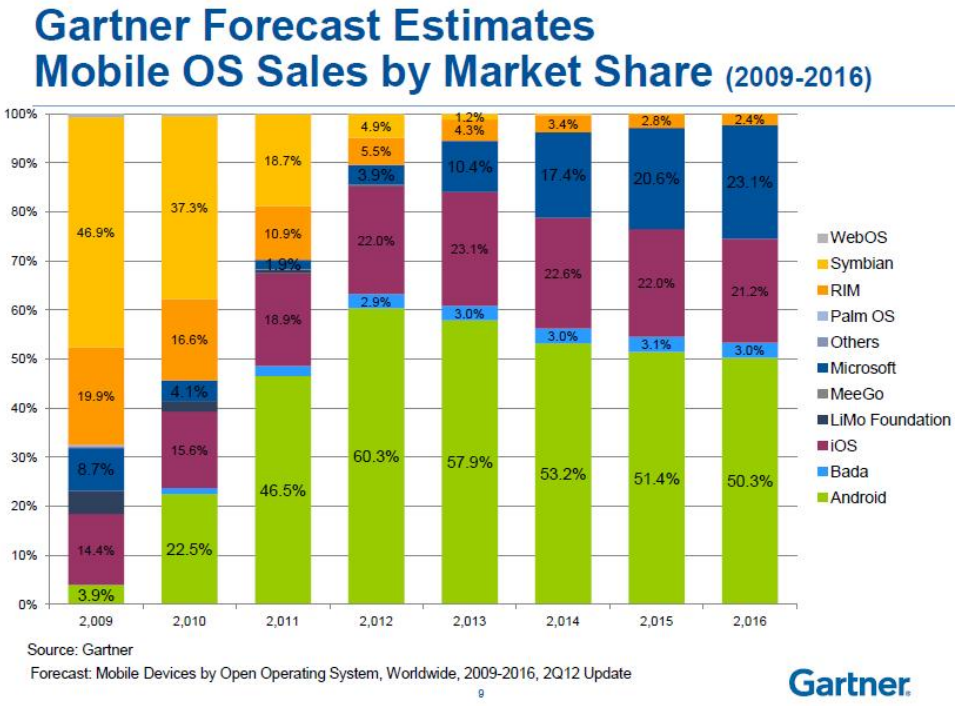
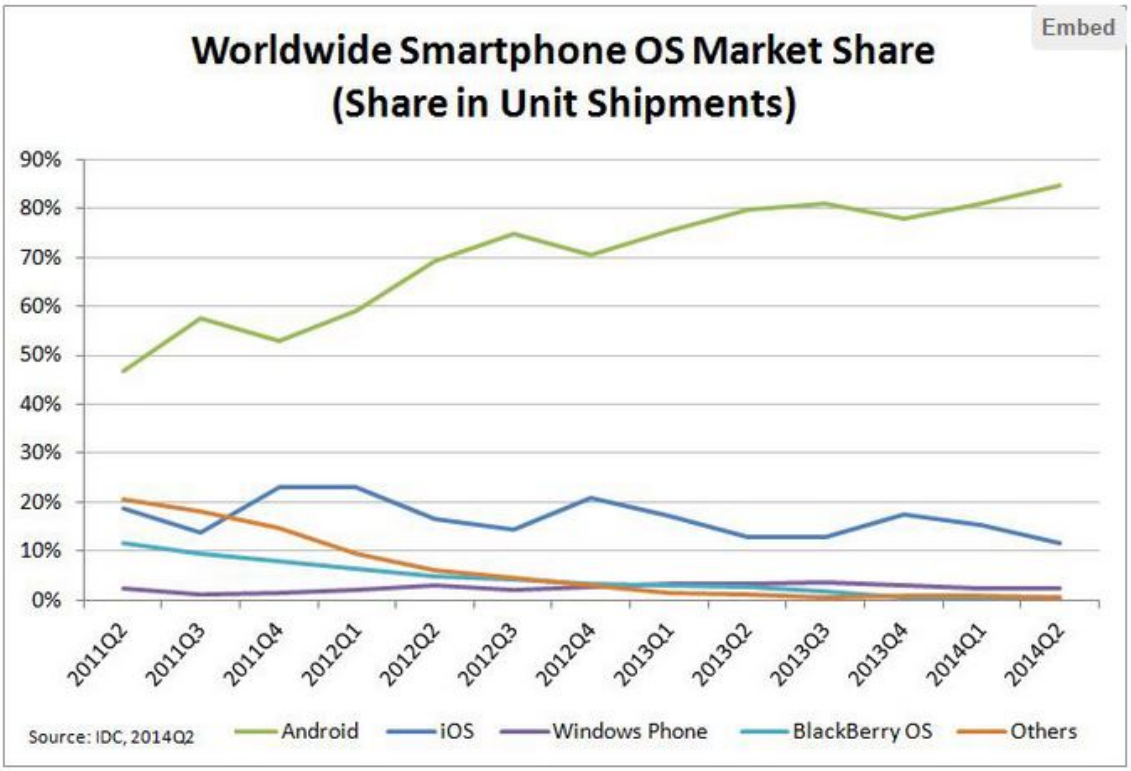


Figura 2: Previsión de mercado de los Sistemas Operativos Móviles según Gartner.

En la Figura 3, se muestra el volumen de ventas y el porcentaje de mercado de los cinco sistemas operativos más populares desde el segundo cuatrimestre del año 2011 hasta el segundo cuatrimestre de 2014.



Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q2 2014	84.7%	11.7%	2.5%	0.5%	0.7%
Q2 2013	79.6%	13.0%	3.4%	2.8%	1.2%
Q2 2012	69.3%	16.6%	3.1%	4.9%	6.1%
Q2 2011	36.1%	18.3%	1.2%	13.6%	30.8%

Source: IDC, 2014 Q2

Figura 3: IDC Top 5 SSOO: Smartphones (IQ 2011-2012-2013-2014)

Como se puede observar el líder indiscutible es el sistema operativo **Android**, presentando un 84.7% de las unidades vendidas en el segundo cuatrimestre de 2014. (IDC, 2014)

Estos datos confirman el increíble crecimiento de este sistema operativo en el mercado, siendo el más usado por los usuarios de los dispositivos móviles actuales, causando que sea el **deseo ideal de los atacantes**.

Características de Android

Android ejecuta las aplicaciones en máquinas virtuales independientes unas de otras, denominadas actualmente *Dalvik* pero que en un futuro realmente próximo se pasarán a llamar *ART*.

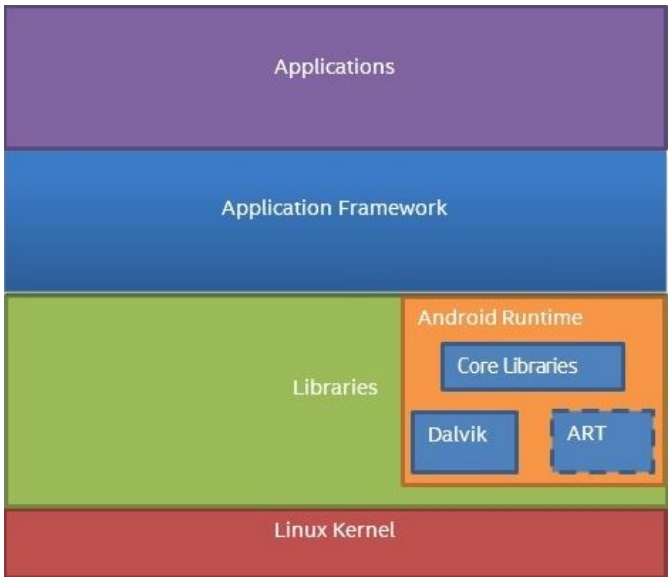


Figura 4: Niveles globales de Android, ART vs Dalvik.

La diferencia básica entre las dos máquinas es que la *Dalvik* es *"Just-In-Time (JIT)"*, es decir, ejecuta el código al tiempo que se inicia la aplicación, y la *ART* es *"Ahead-Of-Time (AOT)"*, es decir, comienza una pre-compilación al instalar la aplicación reduciendo así el tiempo de inicio como se ve en la Figura 5. ((Intel), 2014)

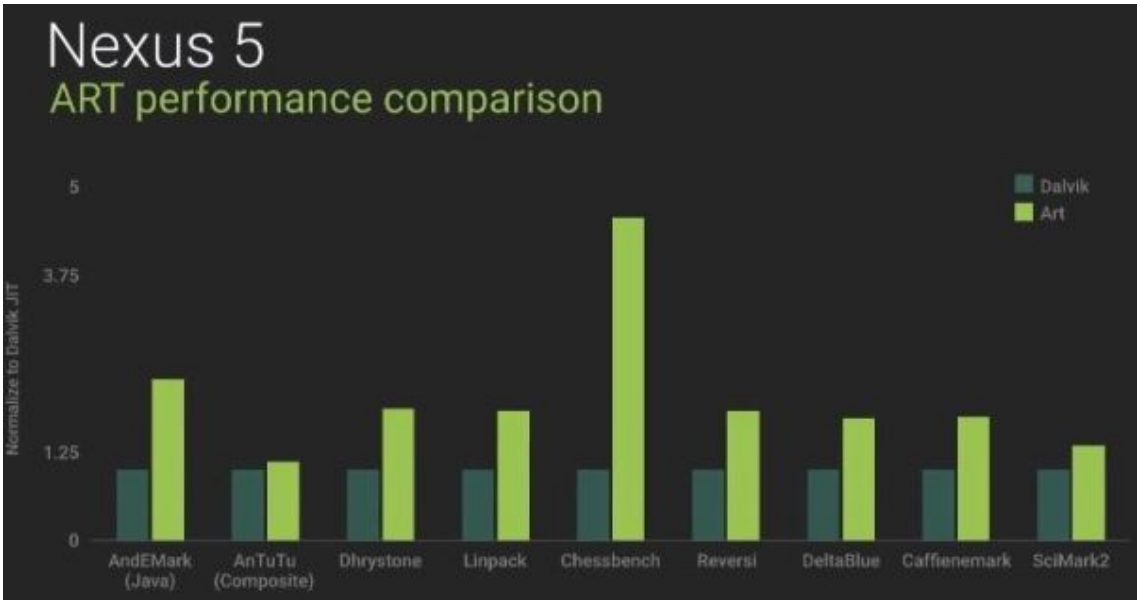


Figura 5: Comparación de rendimiento ART vs Dalvik en un Nexus 5.

Las versiones del sistema operativo Android han ido cambiando cada muy poco tiempo añadiendo en cada nueva versión muchas interesantes y golosas mejoras. Hasta el día de hoy tenemos las siguientes versiones: (Google, s.f.) (Google, Android - Honeycomb, s.f.) (Google, Android - Ice Cream Sandwich, s.f.) (Google, Android - Jelly Bean, s.f.) (Google, Android KitKat, s.f.) (La historia de Android, 2014)

Android 1.0 – Apple Pie: 

Fue la primera versión comercial del software y se lanzó el 23 de septiembre de 2008, pero el primer dispositivo Android que montó esta versión fue el HTC Dream e día 22 de octubre de 2008.

Presenta una interfaz muy sencilla y con pocas opciones de configuración para el usuario.

Android 1.1 – Banana Bread: 

Fue lanzada en febrero de 2009. Ésta versión apenas introdujo cambios, lo que hizo más bien fue ser una versión dedicada a arreglar fallos y bugs de Android 1.0 Apple Pie. La única cosa que añadió fue algo que realmente potenció el sistema operativo, las famosas actualizaciones automáticas.

Android 1.5 – Cupcake: 

Google la lanza en abril de 2009. Se actualizaron la mayoría de los elementos de la interfaz de usuario, se incorporó el teclado táctil desplegable QWERTY, grabador de video, Bluetooth, además de un SDK para desarrollar aplicaciones por parte de terceros.

Android 1.6 – Donut: 

Aparece en septiembre de 2009, añadiendo un buen número de mejoras en la interfaz, y además añadió diversas mejoras en el núcleo, como dar compatibilidad a diferentes resoluciones de pantalla, soporte a VPN y 802.1x…

Android 2.0/2.1 – Eclair: 

La versión 2.0 fue liberada en octubre de 2009 y la versión 2.1 en enero de 2010. Fue adaptada para terminales de mayor tamaño, además de añadir mejoras en el funcionamiento del sistema como el

soporte multicuentas en un mismo terminal, GPS gratuito con Google Maps, soporte HTML5 en el navegador…

Android 2.2 – Froyo: 

Aparece en mayo de 2010, presentando optimizaciones de rendimiento, *tethering*, grabación de video a 720p, soporte Adobe Flash 10.1, una galería rediseñada…

Android 2.3 – Gingerbread: 

Fue lanzada en diciembre de 2010, suponiendo una revolución y llegando a ser la versión de Android más usada en el mundo. Incluye un diseño moderno más estético, soporte con resoluciones más altas y pantallas de mayor tamaño, soporte para conexión NFC, teclado mejorado, soporte para cámaras frontales, sustitución del sistema de archivos YAFFS por el sistema ext4 característico de Linux…

Android 3.x – Honeycomb: 

Se lanzó en febrero de 2011, fue una versión exclusiva para tabletas, y lo único que hizo fue adaptar la interfaz a las tabletas.

Android 4.0 – Ice Cream Sandwich: 

Fue lanzado en octubre de 2011, adaptando la versión de Android 3.0 para Smartphones. Además se añadieron diversas mejoras como la captura de pantalla, soporte MKV, transferencia de datos por NFC, interfaz Holo…

Android 4.1/4.2/4.3 – Jelly Bean: 

La versión 4.1 fue publicada en junio de 2012, la versión 4.2 en noviembre de 2012 y la versión 4.3 en julio de 2013. Es la versión más utilizada en este momento. Introduce gran cantidad de cambios, principalmente de rendimiento y funcionalidad de la interfaz de usuario, además de introducir Google Now, el navegador Google Chrome, streaming de video y audio, captura de fotografías en 360º, mejoras en el soporte multiusuario y multiperfil, compatibilidad con TRIM, Bluetooth Smart, OpenGL ES 3.0…

Android 4.4 – KitKat: 

Aparece en septiembre de 2013, y ha sido la más esperada de todas. Está pensada para ser la versión unificadora de Android y acabar en parte con la fragmentación que había. Las mejoras que incluye son múltiples, como la compatibilidad con terminales desde 512 MB de memoria RAM, reducción en el consumo de la batería, aplicaciones en pantalla completa…

Android L:

Es una versión experimental que introduce múltiples cambios en la interfaz de usuario con los famosos diseños "planos", nuevas notificaciones en la pantalla de bloqueo…

Android Wear:

Es una versión que reafirma que los dispositivos "*weareables*" han llegado para quedarse, por lo que cada vez más se usaran los *smartwatch*, pulseras de monitorización…

Arquitectura de Android

La arquitectura de Android se caracteriza por tener una organización basada en capas. Las cinco capas que se representan en la Figura 6 son: (UC3M, s.f.)

- **Aplicaciones:** este nivel consta de las aplicaciones de usuario, utilizando éstas los niveles inferiores de la arquitectura para su correcto funcionamiento.
- **Framework de Aplicaciones:** este nivel consta de las herramientas de desarrollo de aplicaciones, siendo la mayor parte de ellas bibliotecas Java que acceden a recursos de la máquina virtual *Dalvik*.
- **Librerías:** este nivel está formado por las librerías utilizadas por el sistema operativo Android ofreciendo las capacidades de éste.
- **Entorno de ejecución:** esta capa está al mismo nivel que las librerías. Está formada por las bibliotecas básicas de Android y Java, además de la máquina virtual Dalvik.
- **Kernel de Linux:** esta capa se ocupa de gestionar los recursos hardware que ofrece el terminal a través de llamadas al sistema, como por ejemplo la cámara, audio, memoria, conexiones…

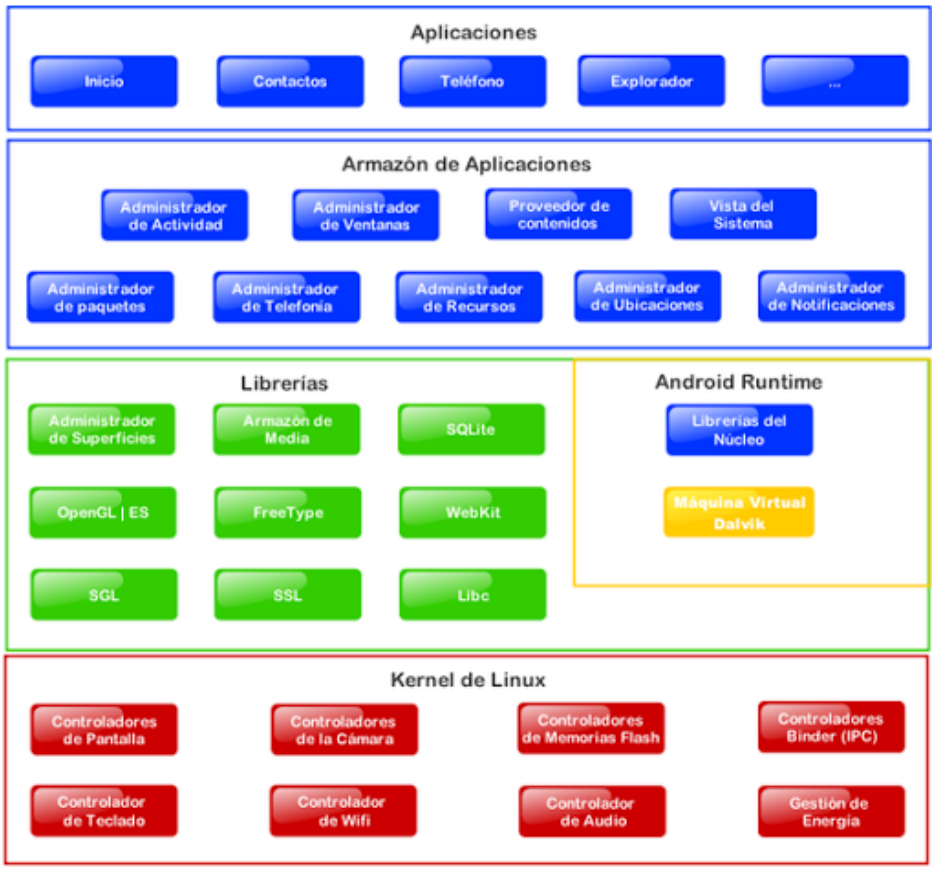


Figura 6: Arquitectura de Android.

Entorno de desarrollo en Android

Para desarrollar en la plataforma Android disponemos de varias opciones: (Google, Android SDK, s.f.) (Google, Android NDK, s.f.)

- SDK: "Software Development Kit" incluye el código fuente, las herramientas y librerías necesarias, además de un emulador para simular el uso de las aplicaciones desarrolladas. Éstas están escritas en Java y corren en la máquina virtual Dalvik/ART.

- NDK: "Native Development Kit" incluye todo lo necesario para implementar las aplicaciones en código nativo como C o C++, lo que nos proporciona reutilización de código y aumento de velocidad al no tener que ejecutar en Dalvik.

El entorno usado para desarrollar este TFG es el SDK como se puede apreciar en la siguiente figura:

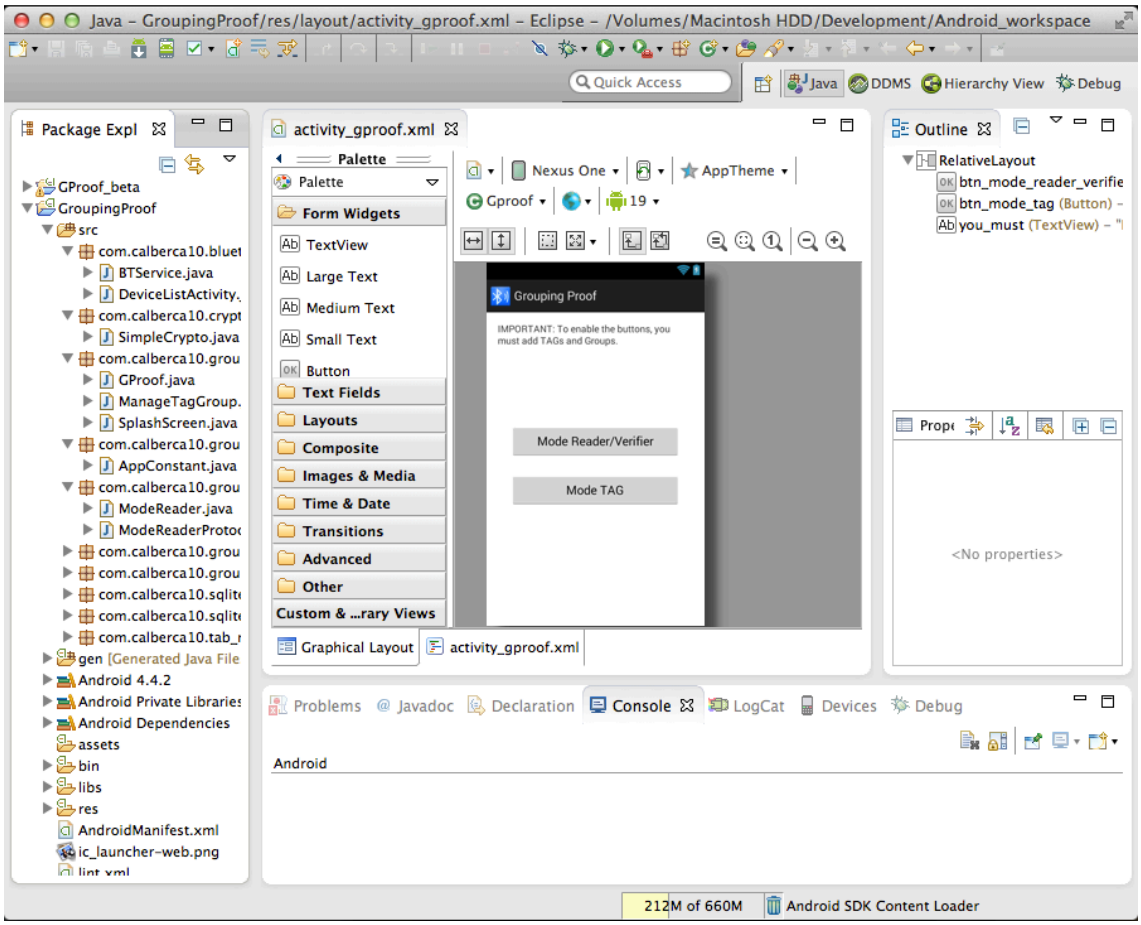


Figura 7: IDE Android.

2.1.2 Arduino

Arduino es una plataforma de hardware abierta o lo que es lo mismo "Hardware Open Source". Está formada por un microcontrolador y un entorno de desarrollo, diseñada para facilitar la utilización de la electrónica en proyectos multidisciplinares. (Arduino, s.f.)

El lenguaje de programación utilizado para esta plataforma es C, de cual existen multitud de tutoriales y librerías que te hacen muy sencilla la implementación para cualquier entorno. Además, una de las cosas más importantes es que la comunidad Arduino ha creado un foro oficial donde se proponen soluciones a los problemas que surgen, que de hecho, para este proyecto me fue de gran ayuda (link).

Existen muchos tipos de placas Arduino, diferenciándose entre sí en la potencia, tamaño, pines de control e incluso aplicaciones (por ejemplo, el modelo LilyPad está encaminada para aplicaciones textiles). Para este TFG se ha hecho uso de un Arduino UNO. A continuación se muestran las características del mismo:

Microcontrolador	ATmega328
Pines digitales E/S	14 (6 de ellos configurables en modo PWM)
Pines analógicos	6
Voltaje de operación	5 V
Voltaje de entrada recomendado	7 – 12 V
Voltaje de entrada	6 – 20 V
Intensidad DC por cada pin E/S	40 mA
Intensidad DC por cada pin 3.3 V	50 mA
Memoria Flash	32 kB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de reloj	16 MHz

Tabla 1: Características Arduino UNO.

La alimentación de la placa se puede realizar mediante la conexión USB, mediante un adaptador AC-DC o mediante pilas. En lo referente a las entradas/salidas, comentar que todas ellas (analógicas y digitales) trabajan a 5V.

La conexión con el ordenador se realiza mediante la comunicación serie UART TTL (5V), de la cual nos informan los leds RX (recibir) y TX (transmitir).

Figura 8: Arduino UNO rev3.

El IDE (Integrated Development Environment) de Arduino es suministrado por el propio fabricante. La versión que se ha usado para este TFG es la 1.5.7 Beta tal y como se puede observar en la siguiente figura:

Figura 9: Arduino IDE 1.5.7.

2.1.3 Bluetooth

El origen de la palabra bluetooth se remonta a cuando Jim Kardach trabajaba en este sistema de comunicación inalámbrica, que unía ordenadores y teléfonos, se acordó de uno de los reyes vikingos, Harald Bluetooth, famoso por unir sus tribus y comer muchos arándanos. Al final se le quedó el nombre y su logotipo adoptaba una de las iniciales del rey.

La plataforma bluetooth es la que se ha usado como conexión entre los distintos dispositivos móviles usados en este TFG. Por parte de Android se usa el dispositivo por defecto bluetooth que tenga el terminal, y por parte de Arduino se ha optado por usar el chip HC-06. El bluetooth con el que trabajamos en esta tecnología es de Clase 1, es decir, que tenemos un rango de unos 10 metros aproximadamente dependiendo de los obstáculos que nos encontremos entre medias.

Módulo Bluetooth HC-06

Este chip va montado sobre una placa JY-MCU para que su uso sea mucho más sencillo, disponiendo de una forma accesible los pines requeridos para su funcionamiento.

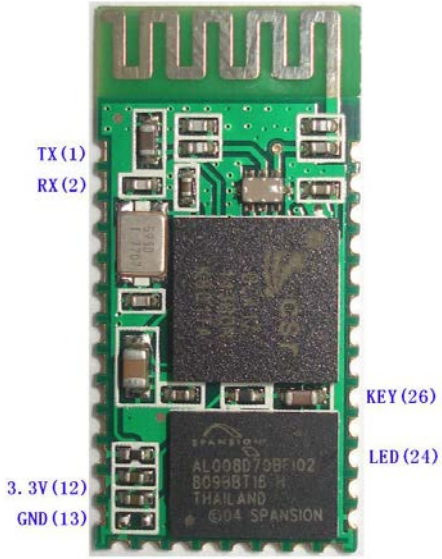


Figura 10: Chip Bluetooth HC-06.

Figura 11: Módulo Bluetooth HC-06 LINVOR (sobre placa JY-MCU).

Este modelo actúa solamente como esclavo y no como master, lo que quiere decir que sólo acepta que se le conecten dispositivos bluetooth, no pudiéndose así conectarse a otros dispositivos por su cuenta.

Comunicación y conexiones requeridas

La conexión mediante la placa Arduino y el módulo bluetooth se realiza a través del puerto serial situado en los pines digitales D0 y D1 del Arduino. Notar que estos pines se pueden cambiar mediante el uso de la librería *"SoftwareSerial.h"*.

A continuación se especifican los pines de los que dispone el módulo bluetooth y que aparecen en la Figura 12: (A CHILD'S GUIDE to BASIC TWO WAY BLUETOOTH COMMUNICATION)

- 5V → Pin por el cual recibe la corriente para su funcionamiento.
- TxD → Pin por el cual transmite los bits necesarios para su comunicación y que va conectado al RxD del Arduino.
- RxD → Pin por el cual recibe los bits necesarios para su comunicación y que va conectado al TxD del Arduino.
- GND → Pin con conexión a tierra.

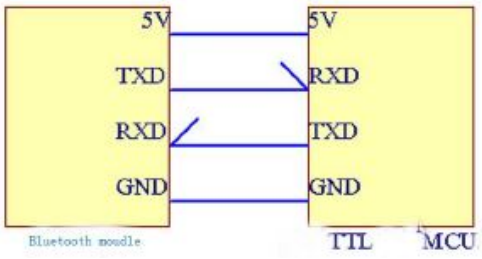


Figura 12: Conexión entre Arduino y Módulo Bluetooth HC-06

El LED

El módulo bluetooth está dotado de un LED de color rojo que indica el estado en el que está. Los diferentes estados son los siguientes:

- Apagado: No hay corriente.
- Parpadeando: Si parpadea a unos 2Hz, lo que indica es que está listo y esperando que algún dispositivo se le conecte.
- Permanente: Indica que la conexión ha sido establecida.

Comandos AT

El módulo bluetooth se configura mediante los famosos comandos AT. Entre comando y comando la espera mínima debe de ser de 1 segundo o más. Los principales comandos son los siguientes: (Guangzhou HC Information Technology Co., 2011)

- Test de comunicación:

Envío: AT Devuelve: OK

- Asignar el "serial baud rate":

Envío: AT+BAUD1 Devuelve: OK1200

Nota: BAUD1 y 1200 son ejemplos de los diferentes valores que se pueden asignar.

- Asignar nuevo Nombre:

Envío: AT+NAMEname Devuelve: OKname

- Asignar nuevo PIN(4bits-number):

Envío: AT+PINxxxx Devuelve: OKxxxx

Características

A continuación se van a nombrar algunas de las características más importantes de este módulo bluetooth: (Guangzhou HC Information Technology Co., 2011)

Sensibilidad	-80dBm
Profundidad de modulación	2Mbps – 3Mbps
Antena	2.4GHz
Medidas	27mm x 13mm x 2mm

Tabla 2: Características Bluetooth HC-06 LINVOR

2.2 CONTEXTO ABARCADO

En este punto se van a tratar los ámbitos en los cuales nos adentramos con este TFG.

2.2.1 Internet of Things / Internet of You

Mucho se está hablando actualmente del "*Internet of Things*" o "*Internet of You*". Se trata de mundos donde todo se encuentra interconectado, como por ejemplo coches, neveras, relojes, etc., los cuales tienen tecnología embebida para interactuar con estados internos o con el ambiente externo. (Cisco, 2014)

Figura 13: Internet of Things

Kevin Ashton, de la empresa Procter & Gamble, en 1999 fue el que por primera vez pronunció el término *"Internet of Things"* para referirse a la conexión con la tecnología **RFID**, fundando el centro Auto-CID del MIT para desarrollar ideas más allá. (Cabezudo, 2014)

IoT está conectando actual gran cantidad de dispositivos a internet, como plantas de fabricación, redes de energía, centros de salud y sistemas de transporte. Cuando un objeto es representado digitalmente, éste puede ser controlado desde cualquier sitio. Esta conectividad significa más cantidad de datos recogidos desde más lugares, con más formas de aumentar la eficiencia y mejorar la seguridad.

A continuación se presenta una imagen que recoge perfectamente un *"timeline"* del IoT: (Cabezudo, 2014)

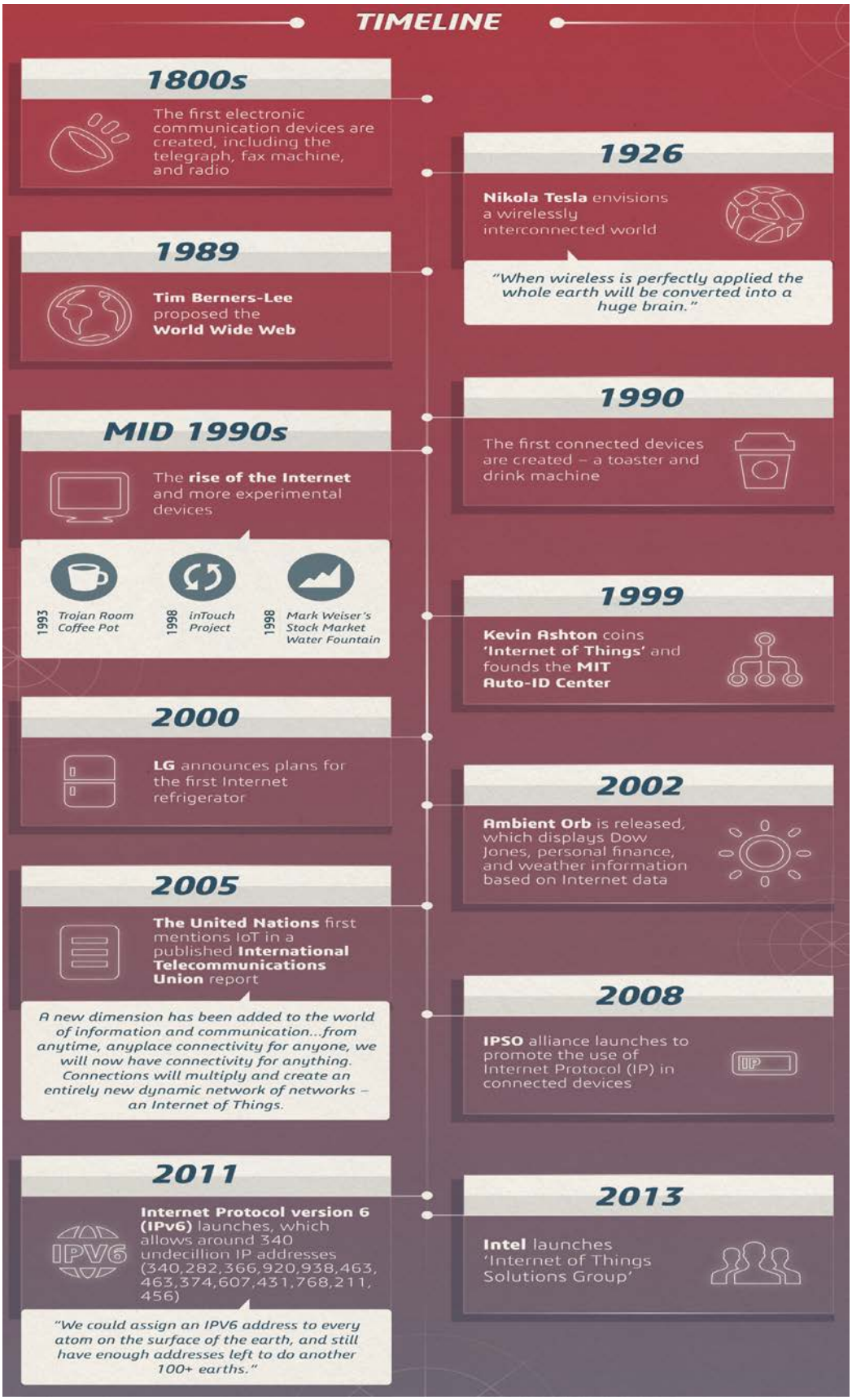


Figura 14: Timeline Internet of Things (English chart).

Mirando hacia el futuro, el IBSG de Cisco calcula que habrá 25 Billones de dispositivos conectados a Internet en 2015 y 50 Billones en 2020. Estas estimaciones no tienen en cuenta los rápidos avances de la tecnología, por lo que la cifra podría ser aún mayor.

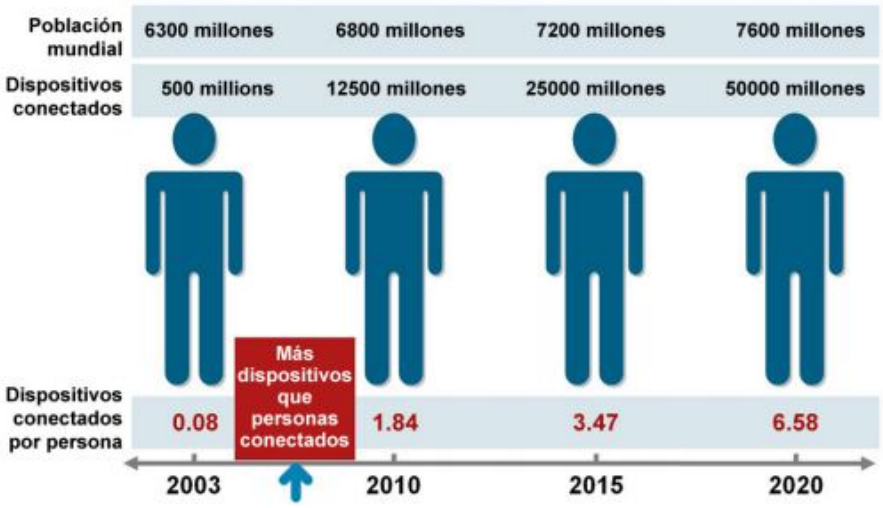


Figura 15: Cantidad de dispositivos, IoT.

En la Figura 15, pueden parecer pocos los dispositivos por persona, pero esa cantidad está calculada con el total de la población mundial, gran parte de la cual todavía no está conectada a internet. (Evans, 2011)

"Con un billón de sensores integrados en el entorno, todos conectados por sistemas informáticos, software y servicios, se podrá escuchar el pulso de la Tierra, lo que tendrá un efecto tan profundo en la interacción humana con el planeta como profunda ha sido la revolución que ha significado Internet para las comunicaciones". Peter Hartwell, Investigador jefe, Laboratorios de HP.

Los principales riesgos tecnológicos, como los regulatorios y legales, que se derivan de los avances tecnológicos y el ecosistema de interconexión son: (Jaquero, 2014)

- **Interferencia en la privacidad:** cuando los dispositivos recogen información acerca de nuestros gustos y preferencias para ser procesada y usada no siempre con la finalidad con que se recaba.
- **Proliferación de tecnologías:** supondrá incrementar el número de fallos al interconectar dispositivos con tecnologías incompatibles, y en consecuencia, un incremento de costes para el fabricante.
- Inexistencia de estándares.

- **Recogida de datos no autorizada:** lo que se conoce como *"Big Data"* es otro reto más a afrontar.

Con todo este mundo tan interconectado, la seguridad y la privacidad de los usuarios son muy importantes: (Jaquero, 2014)

- Todo dispositivo que esté conectado a internet es susceptible de ser accedido, y por lo tanto, deberá de ser protegido.
- Deberían desarrollarse estándares de seguridad de fabricación de dispositivos con conexión a internet para que todos los fabricantes puedan seguirlos.
- Cualquier dispositivo que venga configurado con una *password* por defecto deberá ser cambiada siguiendo unas reglas de complejidad mínimas.
- El fabricante puede diseñar un dispositivo para ser seguro, pero en última instancia dependerá del usuario protegerlo y asegurarse de que es inaccesible; por ejemplo, accediendo a la web del fabricante para descargarse las actualizaciones del dispositivo que cubran vulnerabilidades detectadas.
- Conocer nuestros derechos y obligaciones en relación a la privacidad y protección de nuestros datos personales.

2.2.2 RFID

Cada vez es más frecuente ver tarjetas identificadoras sin contacto con el sistema de lectura. Por ejemplo, recientemente la Comunidad de Madrid ha implantado en su sistema de transporte público las nuevas tarjetas para el abono transporte, que incorporan una etiqueta RFID. Poco a poco, toda la infraestructura está migrando hacia este nuevo sistema, y prácticamente todos los autobuses, paradas de metro, estaciones de tren, etc. de la comunidad están dotadas con un lector.

Figura 16: Tarjeta de Transporte RFID de la Comunidad de Madrid.

Este tipo de sistemas se llaman abreviadamente RFID *"Radio Frequency Identification"* o lo que es lo mismo Identificación por Frecuencia. Estos dispositivos están sustituyendo poco a poco a las etiquetas de códigos de barras y a las tarjetas magnéticas en todas sus aplicaciones.

Figura 17: RFID – "Radio Frequency Identification"

Arquitectura

Un sistema RFID está formado por los siguientes componentes:

- **Etiqueta RFID (transpondedor, TAG):** formada por una antena, un transductor radio y un chip. El chip transmite la información que guarda en la memoria a través de la antena. Hay tres tipos de memoria:
  - Solo lectura: código de identificación único.
  - Lectura y Escritura: la información de identificación puede ser modificada por el lector.
  - Anticolisión: permiten que el lector identifique varias etiquetas a la vez dentro de una mima zona de cobertura.
- **Lector RFID (transceptor, Reader):** formado por una antena, transceptor y un decodificador. El lector envía periódicamente señales para ver si hay una etiqueta dentro de su cobertura. Cuando extrae la información de la etiqueta, se la pasa al subsistema de procesamiento de datos.

- **Subsistema de procesamiento de datos (Middleware RFID, Verifier):** procesa y almacena los datos correspondientes a la lectura de etiquetas. (Toro)

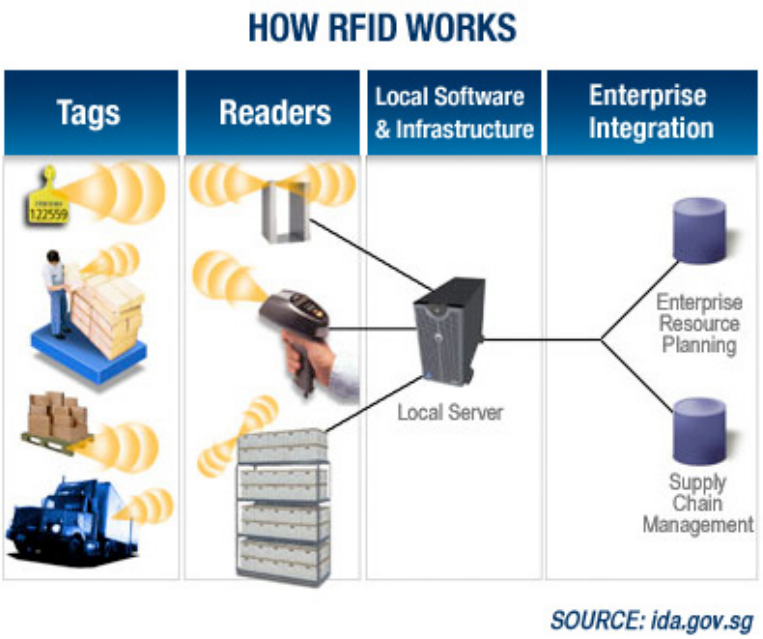


Figura 18: Funcionamiento básico de RFID

Tipos de etiquetas

Existen varios tres tipos de etiquetas: (Wikipedia, 2014)

- **Pasivas:** no requieren ni poseen ningún tipo de alimentación eléctrica, mediante la corriente que les llega con la transmisión de datos, son capaces de generar una respuesta. El rango de funcionamiento es de 6 milímetros a 6 metros.

- **Activas:** poseen su propia fuente autónoma de energía. Son mucho más fiables y tienen menos errores que las etiquetas pasivas. Tienen rangos de milímetros hasta incluso 400 metros.

- **Semipasivas:** también poseen una fuente de alimentación propia, aunque en este caso de usa para alimentar al chip y no para transmitir información. Mantienen una fiabilidad comparable a las activas pero con un rango operativo de las pasivas.

2.2.3 Protocolo "Yoking-Proof"

El protocolo *"Yoking-Proof"* es el protocolo de seguridad que hemos analizado e implementado en este proyecto. Es un protocolo que en un principio está diseñado para los RFIDs pero que nosotros lo hemos implementado en dispositivos móviles realizando la comunicación entre TAGs y Readers (lectores) mediante bluetooth.

Este protocolo, está encaminado a la generación de una evidencia de que dos o más TAGs han sido escaneados simultáneamente (o al menos en un tiempo muy reducido) por un Lector dentro de un mismo rango de cobertura.

El Verificador puede tener dos modos de funcionamiento:

- **Online:** el verificador puede enviar y recibir mensajes de los TAGs, vía Lector, a lo largo de la ejecución del protocolo.
- **Offline:** el verificador sólo puede transmitir retos hacia el Lector.

El modo más interesante es el modo offline, ya que no se necesita la persistente presencia del verificador para generar retos de grupo (yoking-proofs).

A continuación se indican las suposiciones asumidas generalmente para este tipo de retos:

- Los lectores RFID pueden ser potencialmente inseguros. El verificador es el único que es de confianza.
- Los lectores RFID mantienen un registro de las pruebas de cada sesión. Estos registros no pueden ser manipulados por el atacante
- El verificador es una entidad de confianza que pueden compartir alguna información privada con los TAGs como claves criptográficas. El verificador tiene un canal seguro (privado y autenticado) que lo une con los lectores de RFID. En contra, el canal entre los TAGs y el lector es considerado inseguro.
- Para el diseño del protocolo yoking-proofs, lo más importante es la seguridad de la capa del protocolo y no tanto la capa física o de enlace.

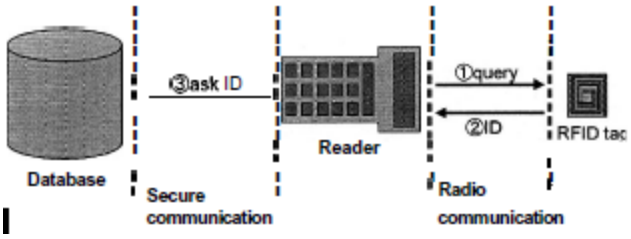


Figura 19: Sistema RFID común.

Hay muchos escenarios importantes de la **vida real** en los que el protocolo **yoking-proofs** puede ser aplicado, ampliando significativamente las capacidades de los sistemas RFID, como por ejemplo estos que se muestran a continuación:

- **Sector farmacéutico:** para demostrar que una medicina fue vendida junto con su receta o con sus instrucciones.
- **Trabajo administrativo del Estado:** para comprobar que un formulario está cerrado con su correspondiente sello o etiqueta.
- **Sistemas de control de acceso:** en las reuniones para generar una evidencia de que un mismo grupo de gente pertenece a una misma ubicación. En los aeropuertos, en los mostradores de facturación, para vincular tu tarjeta de embarque con tu pasaporte y equipajes.
- **Bibliotecas:** en los auto-préstamos para asociar un libro con la tarjeta de identidad del usuario que realiza la adquisición.
- **Entorno familiar:** vincular todos los dispositivos móviles de tus hijos pequeños para alarmarte si alguno de ellos se separa, se pierde, o pierde algún objeto.
- **Masas de gente o zonas peligrosas:** cuando en un futuro todos nuestros objetos lleven chips, se podrá prevenir el robo de nuestras pertenencias, pudiéndote informar en un corto plazo de tiempo o llamar a la policía automáticamente.

El protocolo a implementar para tratar todo este tipo de casos se llama **Kazahaya**, el cual solventa una serie de debilidades de seguridad que fueron encontradas por sus diseñadores. Este protocolo es explicado con más detalle en los próximos capítulos. (Pedro Peris-Lopez, 2011) (Junichiro SAITO, 2005)

CAPÍTULO 3

ANÁLISIS

3.1 PERSPECTIVA GENERAL DEL SISTEMA

En este capítulo se va a describir en detalle el sistema a implementar para garantizar los objetivos definidos en el Capítulo 1 de este documento: desarrollar una aplicación para Android y un programa para Arduino, los cuales, complementándose el uno al otro, sean capaces de completar el protocolo Kazahaya para Yoking-Proof en los distintos escenarios que se han descrito en el punto 2.2.3 de Capítulo 2.

Para ello se analizará dicho protocolo, diviéndolo en funciones y creando un modelo de datos que nos permita su correcto funcionamiento. El programa para la plataforma Arduino dispondrá solamente del modo Etiqueta (TAG), mientras que la aplicación Android dispondrá de dos modos, el modo Etiqueta (TAG) y el modo Lector/Verificador (Reader/Verifier) que se explicarán más adelante.

Los estándares y normas aplicados en los diagramas de este documento serán los acordes con el lenguaje de modelado de sistemas de software UML.

3.2 ALTERNATIVAS DE DISEÑO

En este apartado se va a justificar la elección de las plataformas para desarrollar la aplicación móvil del proyecto, además de la versión de dicha plataforma.

3.2.1 Valoración de las plataformas OS móvil

Como ya analizamos en el punto 2.1.1 del Capítulo 2 (véanse Figura 2 y 3) y gracias al informe de Gartner e IDC, se puede ver que en el mercado actual de los sistemas operativos móviles encontramos unos cuantos con bastante cuota de mercado en 2014, como lo son Android, iOS, Windows Phone y BlackBerry.

Para evaluar y tomar una decisión justificada de las diferentes plataformas con las que nos topamos, se van a analizar las siguientes cualidades:

- Alcance en el mercado. (Ponderación = 35%)
- Velocidad de aprendizaje del lenguaje. (Ponderación = 15%)
- Sencillez de programación. (Ponderación = 20%)
- Soporte. (Ponderación = 30%)

Todas estas cualidades se van a valorar con una puntuación de 1 a 10, siendo 1 la más baja y 10 la más alta, pero la ponderación para alcanzar el 100% es distinta en cada una de ellas tal y como se ha indicado anteriormente.

A continuación se estudian cada una de las cualidades:

Alcance en el mercado

Estudiando la cantidad de dispositivos móviles vendidos conforme a su sistema operativo, se han estipulado los siguientes valores:

Alcance en el Mercado				
	Android	iOS	Windows Phone	BlackBerry
Alcance en el mercado	9	6	4	2
Ponderado (35%)	3.15	2.1	1.4	0.7

Tabla 3: Alcance en el Mercado.

Velocidad de aprendizaje del lenguaje

Evaluando el esfuerzo que hay que hacer para aprender los lenguajes de programación de cada plataforma, se han estipulado los siguientes valores:

Velocidad de aprendizaje del lenguaje				
	Android	iOS	Windows Phone	BlackBerry
Lenguaje	Java	Objetive-C	C#	Java
Alcance en el mercado	9	6	6	9
Ponderado (15%)	1.35	0.75	0.9	1.35

Tabla 4: Velocidad de aprendizaje del lenguaje.

Sencillez de programación

Evaluando la plataforma de desarrollo que hay que usar para implementar las soluciones, se han estipulado los siguientes valores:

Sencillez de programación				
	Android	iOS	Windows Phone	BlackBerry
Sencillez de programación	8	6	6	7
Ponderado (20%)	1.6	1.2	1.2	1.4

Tabla 5: Sencillez de programación.

Soporte de la plataforma

Evaluando el soporte que existe por parte de las empresas y sobretodo desarrolladores, se han estipulado los siguientes valores:

Soporte de la plataforma				
	Android	iOS	Windows Phone	BlackBerry
Soporte de la plataforma	10	8	7	6
Ponderado (20%)	3	2.4	2.1	1.8

Tabla 6: Soporte de la plataforma.

Resumen general de los resultados

A continuación se exponen los resultados generales de las estipulaciones otorgadas anteriormente:

Totales ponderados				
Totales ponderados	**ANDROID**	**iOS**	**Windows Phone**	**BlackBerry**
Alcance en el mercado (35%)	3.15	2.1	1.4	0.7
Velocidad de aprendizaje del lenguaje (15%)	1.35	0.75	0.9	1.35
Sencillez de programación (20%)	1.6	1.2	1.2	1.4
Soporte de la plataforma (30%)	3	2.4	2.1	1.8
TOTAL	9.1	6.45	5.6	5.25

Tabla 7: Resumen de la valoración de las plataformas OS móvil.

Como podemos ver en el TOTAL ponderado, la plataforma que más nota obtiene es Android. Este resultado destaca de esta forma por el gran soporte que tiene dentro de la comunidad de desarrolladores, y además por su alcance en el mercado, sin dejar de tener en cuenta que el desarrollo es en Java (IDE Eclipse), y por lo tanto, bajo el punto de vista del desarrollador del proyecto, no es muy difícil su programación ni su aprendizaje. Debido a ello, la plataforma elegida es ANDROID.

En la Figura 20 que viene a continuación, se ve de una forma rápida y clara los resultados de estas estipulaciones otorgadas:

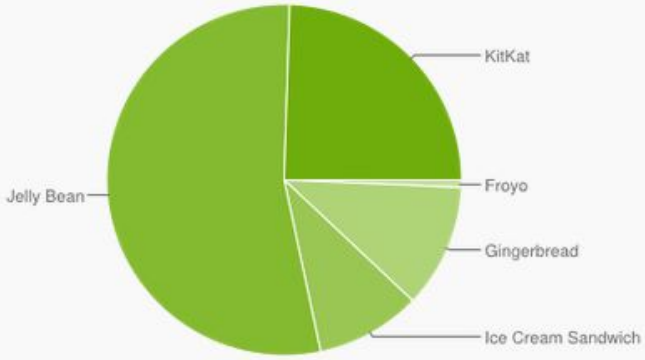
Figura 20: Gráfico con las estipulaciones para las alternativas OS móvil.

3.2.2 Valoración de la versión de Android

Para la elección de la versión de Android en la que desarrollar la aplicación, nos vamos a basar en los datos oficiales que nos proporciona Google del uso de versiones en todos los Smartphones del mundo que ellos controlan.

A continuación mostramos el grafico oficial con el tanto por ciento de Smartphones que hacen uso de cada versión existente de Android:

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%
4.4	KitKat	19	24.5%



Data collected during a 7-day period ending on September 9, 2014.
Any versions with less than 0.1% distribution are not shown.

Figura 21: Porcentaje de dispositivos con cada versión de Android.

Como se puede observar en la Figura 31, las versiones que más extensión tienen son la Jelly Bean y la KitKat, abarcando más del 75% de los Smartphones a día 9 de septiembre de 2014. El rango de APIs a usar en la implementación de la aplicación Android es desde la API 16 (v4.1.X) hasta la API 21(v4.4).

Este uso de APIs no quita que se pueda compilar la aplicación para versiones inferiores puesto que no se hace uso de funciones muy específicas de las versiones más nuevas. Simplemente se han elegido estas APIs para que la interfaz de usuario y el flujo de ventanas sean mucho más minimalista y eficaz.

3.2.3 Valoración Generador de números pseudoaleatorios (PRNG)

Analizando el protocolo Kazahaya *(véase punto 3.3)* nos topamos con el uso de GPAN (generador pseudoaleatorios de números) o PRNG (Pseudo-Random Number Generator), los cuales desempeñan un papel fundamental en la criptografía, puesto que en este campo se requiere que las salidas no sean predichas por salidas anteriores.

Un PRNG no es un número aleatorio como tal porque es completamente determinado por un conjunto de valores iniciales

conocido como *semilla* o *seed (en inglés)*, la cual puede estar formada por números aleatorios. Para su diseño, se requiere de un cuidadoso análisis matemático que de una mínima confianza de que los números son lo suficientemente aleatorios. (Wikipedia, s.f.)

En un principio se pensó en utilizar las funciones que vienen ya implementadas en Java y C (Arduino) para el desarrollo del protocolo. Pero haciendo un pequeño análisis previo, se pudo comprobar que introduciendo la misma semilla en la función PRNG de Java y en la de C, las salidas no eran las mismas. Los PRNG en el protocolo Kazahaya se usan como mecanismos de autenticación (el secreto usado para la autenticación se encuentra encapsulado en la propia semilla) y no como generadores pseudoaleatorios sin más. Por lo tanto, este inconveniente hace que los PRNGs que den distintas salidas dependiendo del entorno de implantación sean descartados. Debido a ello, es una obligación investigar y valorar generadores de números pseudoaleatorios externos a estas plataformas e implementarlos en las mismas. Los elegidos son: (Arduino Tiny, 2012) (Wikipedia, s.f.) (Wikipedia, s.f.)

- ✓ **Mersenne Twister:** es el más utilizado gracias a su calidad. Es desarrollado en 1997 por Makoto Matsumoto y Takuji Nishimura. Su nombre proviene del hecho de que la longitud del periodo corresponde a un *"Número primo de Mersenne"*.
- ✓ **Xorshift:** en julio de 2003 George Marsaglia presenta este generador, pero lo cierto es que no es de gran calidad, aunque es interesante para procesadores pequeños.
- ✓ **JKISS32:** en mayo de 2010 David Jones publica este generador. Consume más SRAM, pero por ejemplo posee mayor calidad que la función aleatoria de Libc.

Para evaluar y tomar una decisión justificada de los diferentes generadores, se van a analizar las siguientes cualidades:

- ➢ Implementación en Arduino ©. (Ponderación = 40%)
- ➢ Implementación en Android (Java). (Ponderación = 40%)
- ➢ Calidad. (Ponderación = 15%)
- ➢ Velocidad en Arduino. (Ponderación = 5%)

Todas estas cualidades se van a valorar con una puntuación de 1 a 10, siendo 1 la más baja y 10 la más alta, pero la ponderación para alcanzar el 100% es distinta en cada una de ellas tal y como se ha indicado anteriormente.

A continuación se estudian cada una de las cualidades:

Implementación en Arduino

Se tiene en cuenta si el generador es capaz de ejecutarse sin ningún problema en Arduino:

Implementación en Arduino			
	Mersenne Twister	**Xorshift**	**JKISS32**
Implementación en Arduino	0	10	10
Ponderado (40%)	0	4	4

Tabla 8: Valoración de PRNGs - Implementación en Arduino.

Implementación en Android

Se tiene en cuenta si el generador es capaz de ejecutarse sin ningún problema en Android:

Implementación en Android			
	Mersenne Twister	**Xorshift**	**JKISS32**
Implementación en Android	10	10	10
Ponderado (40%)	4	4	4

Tabla 9: Valoración de PRNGs - Implementación en Android.

Calidad

Se tiene en cuenta la calidad de los números pseudoaleatorios generados: (Arduino Tiny, 2012)

Calidad			
	Mersenne Twister	**Xorshift**	**JKISS32**
Calidad	10	4	7
Ponderado (15%)	1.5	0.6	1.05

Tabla 10: Valoración de PRNGs - Calidad.

Velocidad en Arduino

Se tiene en cuenta la velocidad de ejecución en la plataforma Arduino (la que menor potencia de procesamiento tiene): (Arduino Tiny, 2012)

Velocidad en Arduino			
	Mersenne Twister	**Xorshift**	**JKISS32**
Velocidad en Arduino	0	8	7
Ponderado (5%)	0	0.4	0.35

Tabla 11: Valoración de PRNGs - Velocidad en Arduino.

Resumen general de los resultados

A continuación se exponen los resultados generales de las estipulaciones otorgadas anteriormente:

Totales ponderados			
Totales ponderados	**Mersenne Twister**	**Xorshift**	**JKISS32**
Implementación en Arduino	0	4	4
Implementación en Android	4	4	4
Calidad (15%)	1.5	0.6	1.05
Velocidad en Arduino (5%)	0	0.4	0.35
TOTAL	5.5	9	9.40

Tabla 12: Valoración de PRNGs - Totales Ponderados.

Como podemos ver en el TOTAL ponderado, los generadores que más nota obtienen son el Xorshift y el JKISS32. Este resultado destaca de esta forma, no por la calidad de los mismos, ya que el generador que más calidad posee es el Mersenne Twister, sino porque son capaces de ejecutarse en Arduino, que realmente es la plataforma que nos pone las limitaciones al ser la de menor potencia de procesado.

Mersenne Twister, a pesar de ser el mejor actualmente, no se ejecuta en Arduino UNO porque no posee de la memoria suficiente para realizar los cálculos correspondientes. Debido a ello, los generadores pseudoaleatorios elegidos son Xorshift y el JKISS32.

En la Figura 32 que viene a continuación, se ve de una forma rápida y clara los resultados de estas estipulaciones otorgadas:

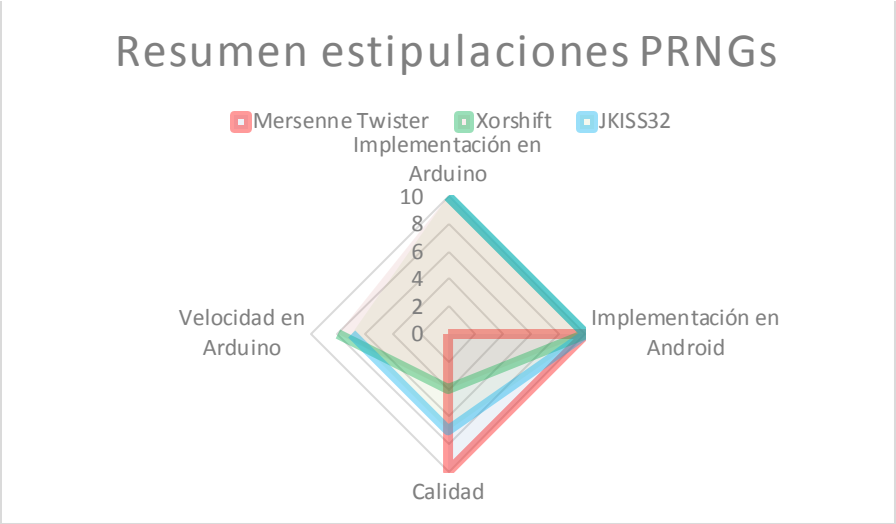


Figura 22: Gráfico con las estipulaciones para las alternativas PRNGs.

3.3 PROTOCOLO KAZAHAYA

En este apartado se va a analizar más técnicamente el protocolo de seguridad que se va a implementar en este TFG, apodado como **Kazahaya** (Yoking-Proof). Pero antes de eso se van a exponer brevemente una serie de consejos prácticos que han sido seguidos para el desarrollo seguro del mismo, para posteriormente explicar el protocolo en sí.

3.3.1 Consejos prácticos seguidos para su desarrollo

Todos los protocolos que habían sido publicados hasta la fecha eran vulnerables a ataques de mayor o menor relevancia. Por ello, para desarrollo de este protocolo se esquematizó una lista con una serie de consejos prácticos que debían ser seguidos por el diseñador del nuevo protocolo para prevenir errores pasados. La lista es la siguiente:

- **Capacidad de Computación:** las etiquetas RFID (tags), en nuestro caso los Arduino UNO tags, son dispositivos dotados de una capacidad de computación reducida. El diseñador del protocolo debe ser consciente de ello para usar unas operaciones que aporten una gran seguridad pero que no sean muy costosas computacionalmente hablando. Por eso se usan PRNG sencillos pero con suficiente aleatoriedad, y operaciones simples como XOR.
- **Dependencia:** cada entrada a una etiqueta (tag) (excepto la primera) debe derivar de las operaciones que se llevan a cabo por etiquetas (tags) participantes en la prueba, garantizando la causalidad de las muestras generadas

durante la prueba. Además de eso, la prueba debe ser ejecutada en un rango de tiempo determinado por el Verificador (Verifier), verificando la causalidad de que las etiquetas (tags) han sido escaneados en la misma ventana de tiempo.

- **Identificación:** la inclusión de números aleatorios parece práctica para construir un identificador con una privacidad más protegida. Aun así, esto no garantiza que el protocolo sea inmune a ataques de privacidad.

- **Emparejando:** introducir un identificador de grupo (ID_Group) y una clave de grupo (K_Group) es importante para probar que los miembros (etiquetas - tags) pertenecen a un mismo grupo. Gracias a esto, las etiquetas (tags) computan que los TAGs que supuestamente pertenecen al grupo, realmente pertenecen, de lo contrario el protocolo se abortaría.

- **Verificación:** se introduce el uso de "*timestamps*" para impedir ataques de repetición. Además se hace uso de encriptación mediante una clave secreta para no hacerlos predecibles. Por lo que se anotarán los "*timestamps*" y los rangos de ventana de tiempo en los cuales cada uno de ellos es válido.

- **Rendimiento:** el número de cálculos al igual que el de mensajes transmitidos por las etiquetas (tags) mediante el canal deben ser mínimos, sin comprometer la seguridad del diseño propuesto. Además los diseñadores, deberían limitar el uso de la memoria NO volátil de las etiquetas (tags) en la cual están almacenados los identificadores y las claves secretas de cada etiqueta (tag).

- **Seguridad "hacia adelante":** Un atacante puede revelar las claves secretas que se encuentren almacenadas en la memoria de las etiquetas (tags), puesto que son susceptibles a la manipulación física. En ciertas situaciones, la seguridad de las comunicaciones anteriores tiene que ser garantizada incluso si una etiqueta (tag) es comprometida más tarde. En un yoking-proof, varias etiquetas (tags) y un Lector no confiable, participan en la generación de pruebas y el Verificador actúa en algún instante posterior, que es lo que se conoce como Evidencia.

3.3.2 Explicación del protocolo

Cabe recordar que este protocolo está encaminado a generar una evidencia de que dos etiquetas (tags) A y B pertenecientes a un mismo grupo fueron escaneadas simultáneamente (o en un tiempo muy reducido) en un mismo sitio. Dicha evidencia quedará almacenada en una BBDD para su posterior verificación por parte del Verificador (Verifier). En este proyecto las funciones de Lector (Reader) y Verificador (Verifier) se han implementado en el mismo dispositivo físico.

Los etiquetas (tags) están divididos en grupos, los cuales están identificados por tener un identificador de grupo (ID_Group), gracias a lo cual prevenimos la participación de etiquetas (tags) no relacionados con la prueba. Además cada etiqueta (tag), que tiene un identificador único, almacena dos claves privadas, una clave de grupo (K_Group) la cual prueba si el miembro pertenece al grupo, y otra clave secreta (K_Tag) que facilita la autenticación de las etiquetas (tags). (Pedro Peris-Lopez, 2011)

Para cada TAG se almacenará en la base de datos la tupla:

$$[ID_{Tag}, ID_{Group}, K_{Tag}, K_{Group}]$$

Los identificadores estáticos (ID_TAG, ID_Group) jamás se enviarán en claro por el canal para garantizar la privacidad. Es más, en cada sesión o pasada de ejecución del protocolo, es decir, desde f1() hasta f7(), se generarán dos números aleatorios (R, R') para cada etiqueta (tag) para prevenir la trazabilidad de las respuestas de cada etiqueta (tag). Para evitar los ataques directos se usará la función XOR.

En la fase de iniciación del protocolo y en las sucesivas pasadas del mismo, el Verifier calcula los "*timestamps*" encriptados por una clave privada del Verifier:

$$TN = E_{Kv}(Timestamp_n)$$

Además de esto, cada TN no es válido para siempre, si no que depende de una ventana de tiempo de validez (Δ), que en nuestro caso es el tiempo por prueba que asignamos a la aplicación. Esta información se guarda en la base de datos mediante la siguiente tupla:

$$[TN, \Delta n]$$

Al finalizar cada pasada del protocolo Kazahaya, se almacenará en la base de datos la tupla de la Evidencia:

$$[ID_{TagA}, ID_{TagB}, TN, R'_a, R'_b, M_{ab}]$$

A continuación se va a presentar un diseño gráfico que explica perfectamente los pasos que realiza el protocolo Kazahaya en una pasada de su ejecución entre dos TAGs y un Reader/Verifier. Pero antes de ello se van a explicar los símbolos que aparecen:

- Operación XOR → ^
- TN → Es el *"timestamp"* cifrado mediante una clave
- PRNG → El generador de números pseudoaleatorios.
- R_x, R_x' → Los números aleatorios.
- Alert() → Función que alerta de que algo ha pasado en el protocolo, como por ejemplo, un ataque, perdida de conexión del TAG, que no pertenecen al mismo grupo…
- GAPs → Espacios de tiempo medidos entre funciones para las pruebas.

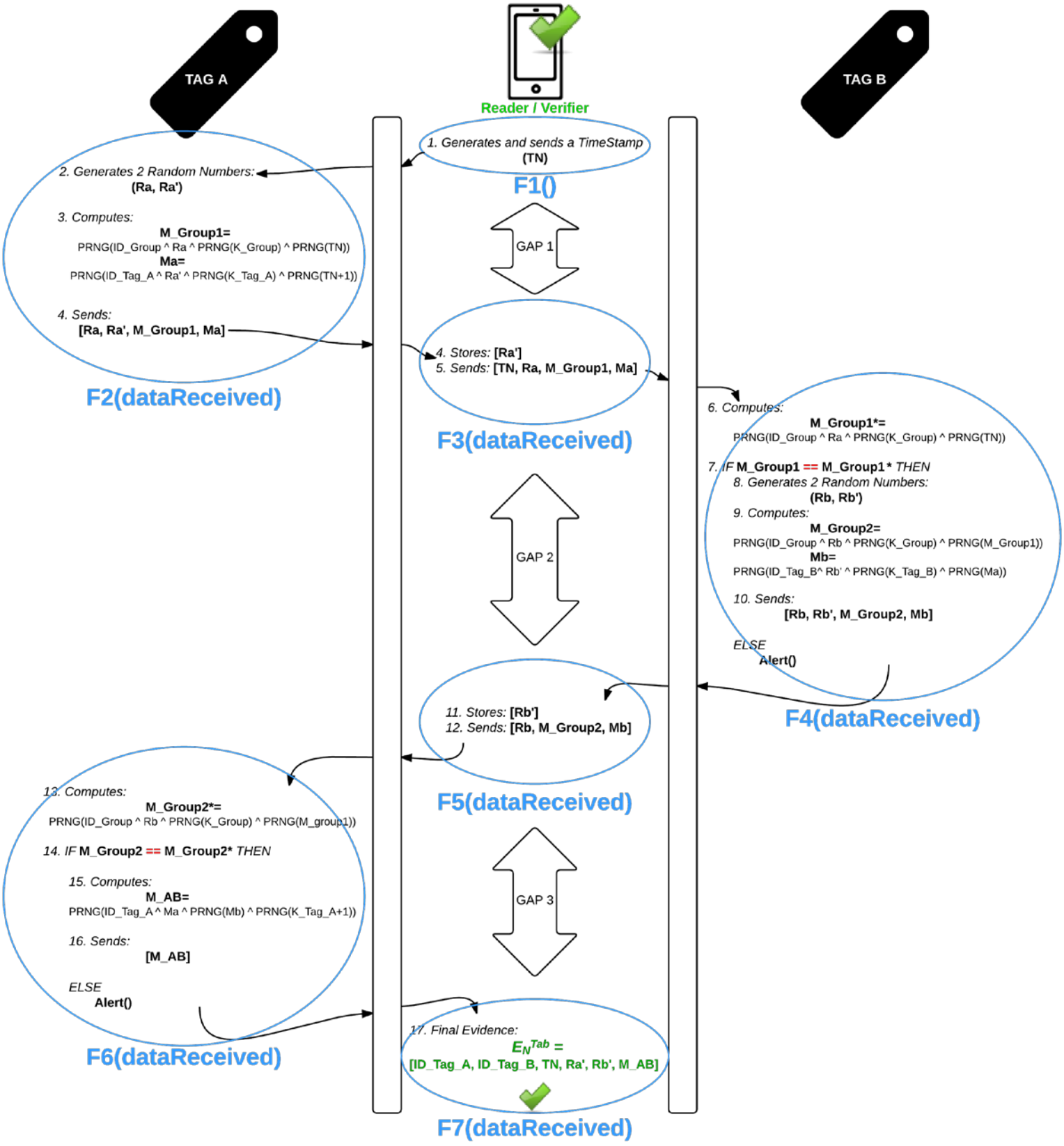


Figura 23: Análisis del protocolo yoking-proof "Kazahaya"

3.4 CASOS DE USO

En este apartado se van a presentar los distintos casos de uso que se pueden dar en la aplicación Android y en el programa Arduino.

3.4.1 Definición de la tabla modelo para los casos de uso

Para definir más en detalle cada caso de uso, se va a usar la siguiente tabla modelo:

ID	CU-Y-XX
Nombre	
Objetivo	
Descripción	
Pre-condiciones	
Post-condiciones	
Escenario	
Causas de fallo	

Tabla 13: Tabla modelo para los Casos de Uso.

Descripción de cada campo de la Tabla 13:

- **ID:** código identificativo único con el formato CU-Y-XX, siendo Y el identificador para saber si se trata de un caso de uso de la aplicación Android (A) o del programa Arduino UNO (AU). Las XX representan el número de caso de uso iniciado en 00.
- **Nombre:** nombre representativo del caso de uso.
- **Objetivo:** definición de la finalidad del caso de uso.
- **Descripción:** presentación de los procesos en cada caso de uso.
- **Pre-condiciones:** condiciones previas para ejecutar el caso de uso.
- **Post-condiciones:** condiciones de salida para una ejecución correcta del caso de uso.
- **Escenario:** relación de pasos que forman el caso de uso.
- **Causas de fallo:** posibles estados que obstaculicen la correcta ejecución del caso de uso.

3.4.2 Casos de uso en la aplicación Android

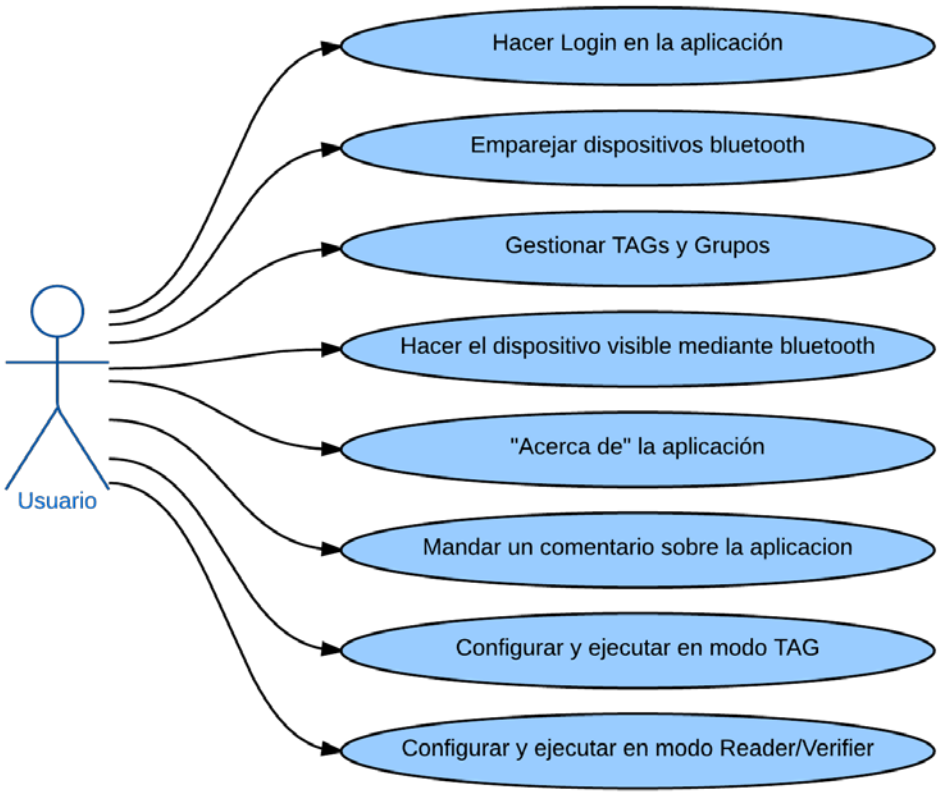


Figura 24: Casos de Uso en la aplicación Android.

Como se puede observar en la Figura 34, el usuario dentro de la aplicación Android puede realizar las siguientes acciones:

- Hacer Login en la aplicación.
- Emparejar dispositivos bluetooth.
- Gestionar TAGs y Grupos.
- Hacer visible el dispositivo mediante bluetooth.
- "Acerca de" la aplicación.
- Mandar un comentario sobre la aplicación.
- Configurar y ejecutar en modo TAG.
- Configurar y ejecutar en modo Reader/Verifier.

A continuación se describe cada caso de uso más técnicamente:

ID	CU-A-01
Nombre	Hacer Login en la aplicación.
Objetivo	Acceder a la aplicación asignando la clave introducida como clave privada del Verifier.
Descripción	El usuario debe introducir la clave secreta que usará el Verifier para cifrar los "timestamps" y así poder acceder al uso de la aplicación.
Pre-condiciones	Introducción de la contraseña en el dialogo mostrado en la aplicación.
Post-condiciones	La aplicación deja entrar al usuario a la aplicación.
Escenario	El usuario introduce una contraseña válida.
Causas de fallo	El usuario deja la contraseña en blanco y pulsa el botón Aceptar. El usuario pulsa el botón Cancelar.

Tabla 14: CU-A-01, Hacer Login en la aplicación.

ID	CU-A-02
Nombre	Emparejar dispositivos bluetooth.
Objetivo	Acceder a la ventana de emparejamiento de dispositivo bluetooth y realizar dicha acción.
Descripción	El usuario quiere escanear dispositivos bluetooth y emparejar los que desee en el caso de que no lo estén.
Pre-condiciones	El bluetooth deberá estar activado.
Post-condiciones	La aplicación informa de los dispositivos escaneados y cuales están emparejados.
Escenario	El usuario pulsa el botón de escanear dispositivos. El usuario pulsa un dispositivo NO emparejado para emparejarlo.
Causas de fallo	El bluetooth no está activado.

Tabla 15: CU-A-02, Emparejar dispositivos bluetooth.

ID	CU-A-03
Nombre	Gestionar TAGs y Grupos.
Objetivo	Acceder a la ventana de gestión de TAGs y Grupos para añadir o consultar éstos.
Descripción	El usuario quiere gestionar (añadir y consultar) TAGs y Grupos, configurando los datos del formulario.
Pre-condiciones	Introducción de todos los campos del formulario.
Post-condiciones	Inserción en la base de datos del nuevo TAG o Grupo.
Escenario	El usuario quiere introducir un nuevo TAG o consultarlos. El usuario quiere introducir un nuevo Grupo o consultarlos.
Causas de fallo	Todos o alguno de los campos del formulario están en blanco.

Tabla 16: CU-A-03, Gestionar TAGs y Grupos.

ID	CU-A-04
Nombre	Hacer visible el dispositivo mediante bluetooth
Objetivo	Hacer que el dispositivo lo puedan ver los demás dispositivos bluetooth por no más de X segundos.
Descripción	El usuario quiere hacer visible el dispositivo para emparejarse con otros dispositivos.
Pre-condiciones	El bluetooth deberá estar activado.
Post-condiciones	El dispositivo tras X segundos deja de estar visible.
Escenario	El usuario pulsa el botón de hacer visible al dispositivo.
Causas de fallo	El bluetooth no está activado.

Tabla 17: CU-A-04, Hacer visible el dispositivo mediante bluetooth.

ID	CU-A-05
Nombre	"Acerca de" la aplicación.
Objetivo	Ir al dialogo que muestra información de la aplicación.
Descripción	El usuario quiere consultar la información de la versión de la aplicación, además del desarrollador y su correo.
Pre-condiciones	El usuario tiene interés en la información de la aplicación.
Post-condiciones	La aplicación te muestra la información de la misma.
Escenario	El usuario pulsa el botón "Acerca de" porque está interesado sobre la aplicación.
Causas de fallo	No corresponde.

Tabla 18: CU-A-05, "Acerca de" la aplicación.

ID	CU-A-06
Nombre	Mandar un comentario sobre la aplicación.
Objetivo	Ir a la aplicación de correo por defecto configurada en el dispositivo.
Descripción	El usuario quiere mandar un correo al desarrollador para informar de algún error de la aplicación o alguna idea de mejora.
Pre-condiciones	El usuario tiene interés en la aplicación.
Post-condiciones	La aplicación te lleva a la aplicación de correo por defecto del sistema.
Escenario	El usuario pulsa el correo del desarrollador porque está interesado en mandarle información útil sobre la aplicación.
Causas de fallo	No hay aplicación de correo instalada.

Tabla 19: CU-A-06, Mandar un comentario sobre la aplicación.

ID	CU-A-07
Nombre	Configurar y ejecutar en modo TAG.
Objetivo	Acceder a la ventana de modo TAG para configurar y ejecutar este modo.
Descripción	El usuario quiere configurar la aplicación en modo TAG para ejercer con ese rol en la simulación del protocolo.
Pre-condiciones	Introducción de todos los campos del formulario de configuración modo TAG.
Post-condiciones	La aplicación te lleva a la ventana de ejecución del protocolo con el rol TAG.
Escenario	El usuario rellena de manera correcta todos los campos del formulario.
Causas de fallo	El usuario no rellena todos los campos del formulario. No se encuentran o no es posible conectar con los dispositivos.

Tabla 20: CU-A-07, Configurar y ejecutar en modo TAG.

ID	CU-A-08
Nombre	Configurar y ejecutar en modo Reader/Verifier.
Objetivo	Acceder a la ventana de modo Reader/Verifier para configurar y ejecutar este modo.
Descripción	El usuario quiere configurar la aplicación en modo Reader/Verifier para ejercer con ese rol en la simulación del protocolo.
Pre-condiciones	Introducción de todos los campos del formulario de configuración modo Reader/Verifier.
Post-condiciones	La aplicación te lleva a la ventana de ejecución del protocolo con el rol Reader/Verifier.
Escenario	El usuario rellena de manera correcta todos los campos del formulario.
Causas de fallo	El usuario no rellena todos los campos del formulario. No se encuentran o no es posible conectar con los dispositivos.

Tabla 21: CU-A-08, Configurar y ejecutar en modo Reader/Verifier.

3.4.3 Casos de uso en el programa Arduino

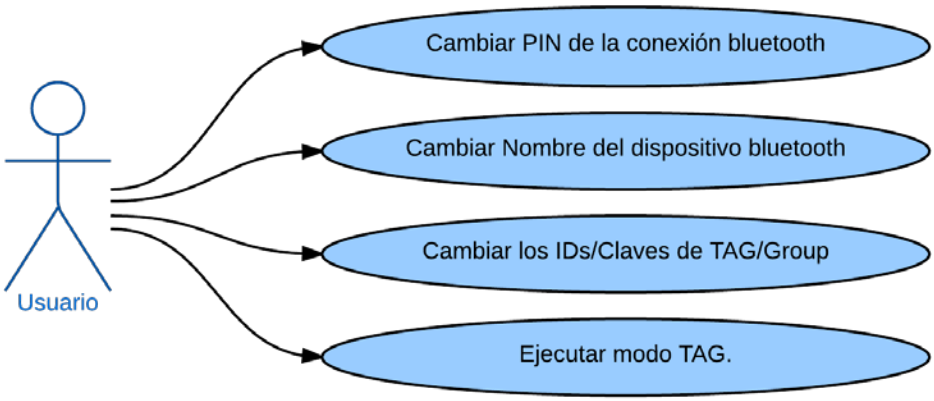


Figura 25: Casos de Uso en la aplicación Arduino.

Como se puede observar en la Figura 35, el usuario dentro del programa Arduino puede realizar las siguientes acciones:

- Cambiar PIN de la conexión bluetooth.
- Cambiar Nombre del dispositivo bluetooth.
- Cambiar los IDs/Claves de TAG/Group.
- Ejecutar modo TAG.

A continuación se describe cada caso de uso más técnicamente:

ID	CU-AU-01
Nombre	Cambiar PIN de la conexión bluetooth.
Objetivo	Reconfigurar el PIN de establecimiento de conexión del dispositivo bluetooth.
Descripción	El usuario quiere cambiar el PIN del dispositivo bluetooth.
Pre-condiciones	Se debe cambiar la configuración del bluetooth a 1 para proceder al cambio, y asignar el nuevo PIN.
Post-condiciones	El dispositivo bluetooth tiene un nuevo PIN de conexión.
Escenario	El usuario rellena correctamente la variable que asigne el nuevo PIN.
Causas de fallo	El usuario no rellena la variable correspondiente. El usuario no activa la reconfiguración del dispositivo bluetooth. El usuario introduce más de 4 caracteres como PIN.

Tabla 22: CU-AU-01, Cambiar PIN de la conexión bluetooth.

ID	CU-AU-02
Nombre	Cambiar Nombre del dispositivo bluetooth.
Objetivo	Reconfigurar el Nombre del dispositivo bluetooth.
Descripción	El usuario quiere cambiar el Nombre del dispositivo bluetooth.
Pre-condiciones	Se debe cambiar la configuración del bluetooth a 1 para proceder al cambio, y asignar el nuevo Nombre.
Post-condiciones	El dispositivo bluetooth tiene un nuevo Nombre de conexión.
Escenario	El usuario rellena correctamente la variable que asigne el nuevo Nombre.
Causas de fallo	El usuario no rellena la variable correspondiente. El usuario no activa la reconfiguración del dispositivo bluetooth. El usuario introduce más de 20 caracteres como Nombre.

Tabla 23: CU-AU-02, Cambiar Nombre del dispositivo bluetooth.

ID	CU-AU-03
Nombre	Cambiar los IDs/Claves de TAG/Group.
Objetivo	Reconfigurar los identificadores o claves, del TAG o del Grupo al que pertenece el TAG.
Descripción	El usuario quiere cambiar el identificador o clave del TAG o del Grupo al que pertenece.
Pre-condiciones	Asignar los nuevos identificadores o claves a las variables correspondientes.
Post-condiciones	El TAG tiene nuevo identificador o clave, y además ha podido cambiar de Grupo.
Escenario	El usuario rellena correctamente las variables correspondientes que asignen los nuevos identificadores o claves.
Causas de fallo	El usuario no rellena las variables correspondientes. El usuario introduce un número más grande que el rango. **Rango** → [0 - 4294967295 (2^{32})] El usuario introduce otro carácter que no sea un número.

Tabla 24: CU-AU-03, Cambiar los IDs/Claves de TAG/Group.

NOTA: El rango del caso de uso CU-AU-03 viene dato debido a las limitaciones del Arduino.

ID	CU-AU-04
Nombre	Ejecutar modo TAG.
Objetivo	Ejecutar el protocolo en modo TAG.
Descripción	El usuario quiere ejecutar el protocolo en modo TAG.
Pre-condiciones	Se debe tener todos los parámetros del protocolo y de conexión bluetooth rellenos.
Post-condiciones	El Arduino ejecuta el modo TAG del protocolo.
Escenario	El usuario rellena correctamente todos los parámetros de configuración y ejecuta el modo TAG.
Causas de fallo	El usuario no rellena los parámetros de configuración correctamente.

Tabla 25: CU-AU-04, Ejecutar modo TAG.

3.5 REQUISITOS

En este apartado se exponen los requisitos de software de la aplicación Android y del programa de Arduino UNO. Haremos una división típica de los mismos: Requisitos de Usuario, Requisitos Funcionales y Requisitos no Funcionales.

3.5.1 Definición de la tabla modelo para los requisitos

Para definir más en detalle cada caso de uso, se va a usar la siguiente tabla modelo:

ID	RZ-Y-XX
Título	
Descripción	
Prioridad	☐Alta ☐Media ☐Baja
Estabilidad	☐Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☐Baja
Verificabilidad	☐Alta ☐Media ☐Baja
Trazabilidad	

Tabla 26: Tabla modelo para los requisitos.

Descripción de cada campo de la Tabla 23:

- **ID:** código identificativo único con el formato RZ-Y-XX. La Z representa si son de usuario (U), funcionales (F) o no funcionales (NF). La Y el identificador para saber si se trata de un caso de uso de la aplicación Android (A) o del programa Arduino UNO (AU). Las XX representan el número de caso de uso iniciado en 00.

- **Título:** título representativo del requisito.
- **Descripción:** definición del requisito.
- **Prioridad:** condición de implementación del requisito. [Alta ; Media ; Baja]
- **Estabilidad:** especifica la sensibilidad del requisito a ser modificado. [Alta ; Media ; Baja]
- **Dificultad:** dificultad de implementación del requisito. [Alta ; Media ; Baja]
- **Verificabilidad:** simplicidad para verificar este requisito. [Alta ; Media ; Baja]
- **Trazabilidad:** trazabilidad a casos de uso.

3.5.2 Requisitos de Usuario aplicación Android

En esta sección se van a definir los requisitos de usuario de la aplicación Android:

ID	RU-A-01
Título	Aplicar contraseña Verificador.
Descripción	El usuario debe ser capaz de introducir una contraseña para asignarla al Verifier. Dicha contraseña se usará para cifrar datos durante el protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-01, CU-A-07, CU-A-08

Tabla 27: RU-A-01; Aplicar contraseña Verificador.

ID	RU-A-02
Título	Activar Bluetooth.
Descripción	El usuario debe ser capaz de activar el bluetooth para el correcto funcionamiento de la aplicación.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-07, CU-A-08

Tabla 28: RU-A-02; Activar Bluetooth.

ID	RU-A-03
Título	Buscar dispositivos bluetooth.
Descripción	El usuario debe ser capaz de buscar dispositivos bluetooth.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-07, CU-A-08

Tabla 29: RU-A-03; Buscar dispositivos bluetooth.

ID	RU-A-04
Título	Conectar dispositivos bluetooth.
Descripción	El usuario debe ser capaz de conectar dispositivos bluetooth.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-07, CU-A-08

Tabla 30: RU-A-04; Conectar dispositivos bluetooth.

ID	RU-A-05
Título	Emparejar dispositivos bluetooth.
Descripción	El usuario debe ser capaz de emparejar dispositivos bluetooth.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-07, CU-A-08

Tabla 31: RU-A-05; Emparejar dispositivos bluetooth.

ID	RU-A-06
Título	Añadir nuevos TAGs.
Descripción	El usuario debe ser capaz de añadir nuevos TAGs.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-03, CU-A-07, CU-A-08

Tabla 32: RU-A-06; Añadir nuevos TAGs.

ID	RU-A-07
Título	Consultar los TAGs existentes.
Descripción	El usuario debe ser capaz de consultar los TAGs existentes.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-03, CU-A-07, CU-A-08

Tabla 33: RU-A-07; Consultar los TAGs existentes.

ID	RU-A-08
Título	Añadir nuevos Grupos.
Descripción	El usuario debe ser capaz de añadir nuevos Grupos.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-03, CU-A-07, CU-A-08

Tabla 34: RU-A-08; Añadir nuevos Grupos.

ID	RU-A-09
Título	Consultar los Grupos existentes.
Descripción	El usuario debe ser capaz de consultar los Grupos existentes.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-03, CU-A-07, CU-A-08

Tabla 35: RU-A-09; Consultar los Grupos existentes.

ID	RU-A-10
Título	Hacer dispositivo visible.
Descripción	El usuario debe ser capaz de hacer el dispositivo visible para poder emparejarse con otros dispositivos bluetooth.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04.

Tabla 36: RU-A-10; Hacer dispositivo visible.

ID	RU-A-11
Título	Consultar "Acerca de" la aplicación.
Descripción	El usuario debe ser capaz de consultar información dentro de la pestaña "Acerca de" del menú.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-05, CU-A-06.

Tabla 37: RU-A-11; Consultar Acerca de la aplicación.

ID	RU-A-12
Título	Enviar correo al desarrollador de la aplicación.
Descripción	El usuario debe ser capaz de enviar un correo al desarrollador de la aplicación para consulta o descripción de error.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-05, CU-A-06.

Tabla 38: RU-A-12; Enviar correo al desarrollador de la aplicación.

ID	RU-A-13
Título	Configurar el modo TAG.
Descripción	El usuario debe ser capaz de configurar el móvil en modo TAG para el uso del protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-07.

Tabla 39: RU-A-13; Configurar el modo TAG.

ID	RU-A-14
Título	Ejecutar el modo TAG.
Descripción	El usuario debe ser capaz de ejecutar el modo TAG en el móvil para el uso del protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-07.

Tabla 40: RU-A-14; Ejecutar el modo TAG.

ID	RU-A-15
Título	Configurar el modo Reader/Verifier.
Descripción	El usuario debe ser capaz de configurar el móvil en modo Reader/Verifier para el uso del protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-08.

Tabla 41: RU-A-15; Configurar el modo Reader/Verifier.

ID	RU-A-16
Título	Ejecutar el modo Reader/Verifier.
Descripción	El usuario debe ser capaz de ejecutar el móvil en modo Reader/Verifier para el uso del protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-A-02, CU-A-04, CU-A-08.

Tabla 42: RU-A-16; Ejecutar el modo Reader/Verifier.

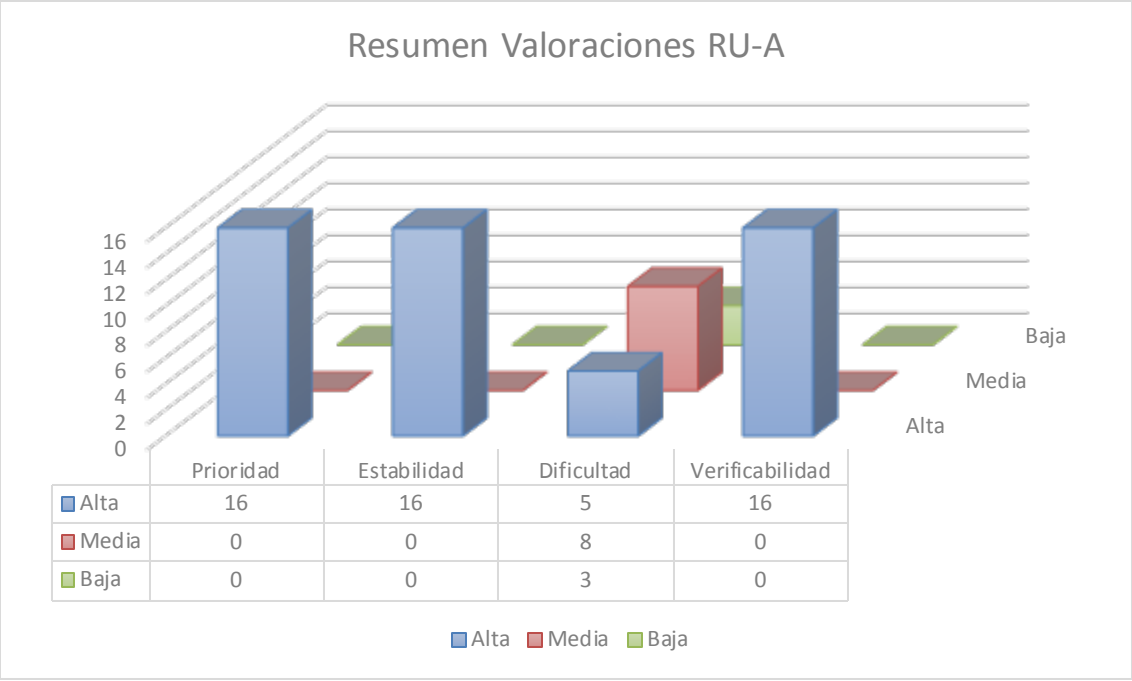


Figura 26: Resumen valoraciones requisitos de usuario aplicación Android.

RU-A/CU-A	CU-A-01	CU-A-02	CU-A-03	CU-A-04	CU-A-05	CU-A-06	CU-A-07	CU-A-08
RU-A-01	X						X	X
RU-A-02		X		X			X	X
RU-A-03		X		X			X	X
RU-A-04		X		X			X	X
RU-A-05		X		X			X	X
RU-A-06			X				X	X
RU-A-07			X				X	X
RU-A-08			X				X	X
RU-A-09			X				X	X
RU-A-10		X		X				
RU-A-11					X	X		
RU-A-12					X	X		
RU-A-13		X		X			X	
RU-A-14		X		X			X	
RU-A-15		X		X				X
RU-A-16		X		X				X

Tabla 43: Matriz de Trazabilidad RU-A / CU-A.

3.5.3 Requisitos de Usuario programa Arduino

En esta sección se van a definir los requisitos de usuario del programa Arduino Uno:

ID	RU-AU-01
Título	Cambiar PIN dispositivo bluetooth.
Descripción	El usuario debe ser capaz de cambiar el PIN del dispositivo bluetooth para hacer nuevas conexiones.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-AU-01.

Tabla 44: RU-AU-01; Cambiar PIN dispositivo bluetooth.

ID	RU-AU-02
Título	Cambiar Nombre dispositivo bluetooth.
Descripción	El usuario debe ser capaz de cambiar el Nombre del dispositivo bluetooth para hacer nuevas conexiones.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-AU-02.

Tabla 45: RU-AU-02; Cambiar Nombre dispositivo bluetooth.

ID	RU-AU-03
Título	Capaz de establecer conexión.
Descripción	El usuario debe ser permitir al dispositivo Arduino aceptar conexiones entrantes para las conexiones bluetooth.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-AU-01, CU-AU-02, CU-AU-04.

Tabla 46: RU-AU-03; Capaz de establecer conexión.

ID	RU-AU-04
Título	Gestionar identificadores de TAG y de Grupo.
Descripción	El usuario debe ser capaz de gestionar los identificadores de TAG o de Grupo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-AU-03, CU-AU-04.

Tabla 47: RU-AU-04; Gestionar identificadores de TAG y de Grupo.

ID	RU-AU-05
Título	Gestionar claves de TAG y de Grupo.
Descripción	El usuario debe ser capaz de gestionar las claves de TAG o de Grupo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-AU-03, CU-AU-04.

Tabla 48: RU-AU-05; Gestionar claves de TAG y de Grupo.

ID	RU-AU-06
Título	Ejecutar modo TAG.
Descripción	El usuario debe ser capaz de ejecutar el protocolo modo TAG.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	CU-AU-01, CU-AU-02, CU-AU-03, CU-AU-04.

Tabla 49: RU-AU-06; Ejecutar modo TAG.

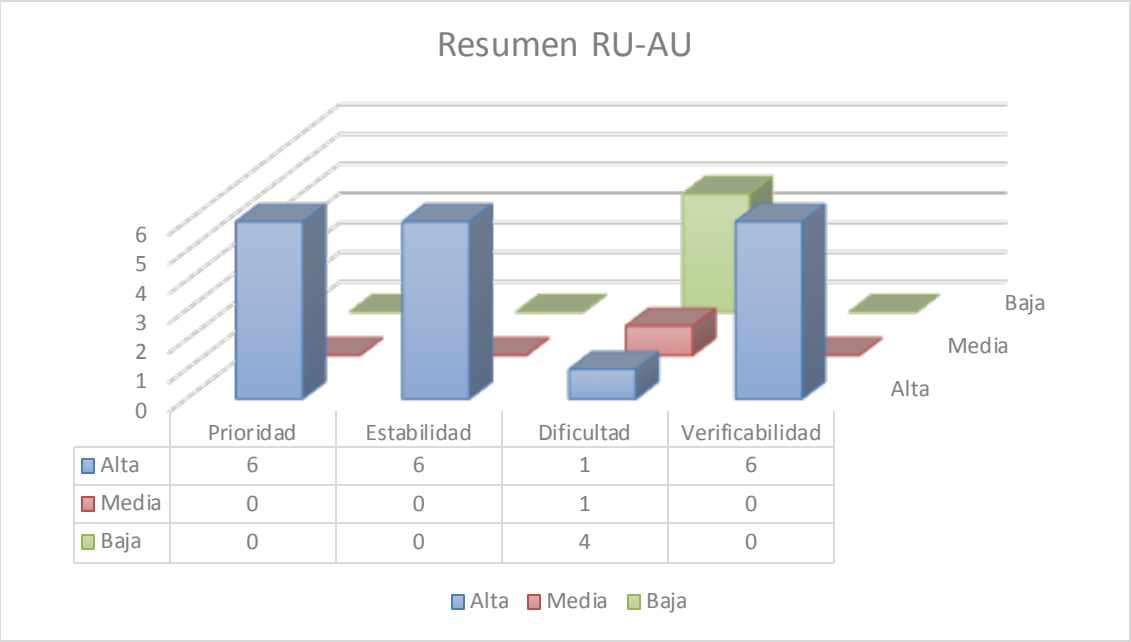


Figura 27: Resumen valoraciones requisitos de usuario programa Arduino.

RU-AU/CU-AU	CU-AU-01	CU-AU-02	CU-AU-03	CU-AU-04
RU-AU-01	X			
RU-AU-02		X		
RU-AU-03	X	X		X
RU-AU-04			X	X
RU-AU-05			X	X
RU-AU-06	X	X	X	X

Tabla 50: Matriz de Trazabilidad RU-AU / CU-AU.

3.5.4 Requisitos Funcionales aplicación Android

En esta sección se van a definir los requisitos de software funcionales del sistema (aplicación Android):

ID	RF-A-01
Título	Interfaz de usuario en inglés o castellano.
Descripción	El sistema debe asignar el idioma según el idioma por defecto en Android.
Prioridad	☐Alta ☒Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-A-XX.

Tabla 51: RF-A-01; Interfaz de usuario en inglés o castellano.

ID	RF-A-02
Título	Contraseña del Verificador alfanumérica de 40 caracteres.
Descripción	El sistema debe permitir una contraseña alfanumérica de máximo 40 caracteres para el cifrado del Verificador.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-01, RU-A-15, RU-A-16.

Tabla 52: RF-A-02; Contraseña del Verificador alfanumérica de 40 caracteres.

ID	RF-A-03
Título	Activar bluetooth al iniciar app o ventanas principales.
Descripción	El sistema debe verificar el estado del bluetooth entre las ventanas principales de la aplicación y al iniciar la app, y si no está activado, preguntar al usuario si activarlo y continuar) o desactivarlo y salir de la aplicación.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-02, RU-A-03, RU-A-04, RU-A-05, RU-A-10, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 53: RF-A-03; Activar bluetooth al iniciar app o ventanas principales.

ID	RF-A-04
Título	Buscar dispositivos bluetooth.
Descripción	El sistema debe listar todos los dispositivos bluetooth al alcance.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-02, RU-A-03, RU-A-04, RU-A-05.

Tabla 54: RF-A-04; Buscar dispositivos bluetooth.

ID	RF-A-05
Título	Conectar dispositivos bluetooth.
Descripción	El sistema debe conectar los dispositivos listados que ya estén emparejados.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-02, RU-A-03, RU-A-04, RU-A-05, RU-A-15, RU-A-16.

Tabla 55: RF-A-05; Conectar dispositivos bluetooth.

ID	RF-A-06
Título	Emparejar dispositivos bluetooth.
Descripción	El sistema debe emparejar los dispositivos listados que no estén ya emparejados.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-02, RU-A-03, RU-A-04, RU-A-05, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 56: RF-A-06; Emparejar dispositivos bluetooth.

ID	RF-A-07
Título	Añadir nuevos TAGs.
Descripción	El sistema debe añadir a la base de datos nuevos TAGs cuando el formulario (Nombre, ID, Clave) esté relleno.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-06, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 57: RF-A-07; Añadir nuevos TAGs.

ID	RF-A-08
Título	Listar TAGs ya añadidos.
Descripción	El sistema debe listar en un *"spinner"* los TAGs que se encuentren en la base de datos.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-07, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 58: RF-A-08; Listar TAGs ya añadidos.

ID	RF-A-09
Título	Añadir nuevos Grupos.
Descripción	El sistema debe añadir a la base de datos nuevos Grupos cuando el formulario (Nombre, ID, Clave) esté relleno.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-08, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 59: RF-A-09; Añadir nuevos Grupos.

ID	RF-A-10
Título	Listar Grupos ya añadidos.
Descripción	El sistema debe listar en un *"spinner"* los Grupos que se encuentren en la base de datos.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-09, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 60: RF-A-10; Listar Grupos ya añadidos.

ID	RF-A-11
Título	El campo Nombre es alfanumérico.
Descripción	El campo Nombre del formulario para añadir TAGs/Grupos tiene que ser alfanumérico con un máximo de 30 caracteres.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-06, RU-A-08, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 61: RF-A-11; El campo Nombre es alfanumérico.

ID	RF-A-12
Título	El campo ID (identificador) es numérico.
Descripción	El campo ID del formulario para añadir TAGs/Grupos tiene que ser numérico con un máximo de 9 caracteres, debido a la operación XOR y la capacidad del Arduino (unsigned long de 32 bits).
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-06, RU-A-08, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 62: RF-A-12; El campo ID (identificador) es numérico.

ID	RF-A-13
Título	El campo Clave es numérico.
Descripción	El campo Clave del formulario para añadir TAGs/Grupos tiene que ser numérico con un máximo de 9 caracteres, debido a la operación XOR y la capacidad del Arduino (unsigned long de 32 bits).
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-06, RU-A-08, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 63: RF-A-13; El campo Clave es numéric.

ID	RF-A-14
Título	Hacer dispositivo visible.
Descripción	El sistema debe permitir visibilizar el dispositivo bluetooth para poder emparejarse.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-03, RU-A-04, RU-A-05, RU-A-13, RU-A-15.

Tabla 64: RF-A-14; Hacer dispositivo visible.

ID	RF-A-15
Título	Mostrar información app.
Descripción	El sistema debe mostrar información básica de la aplicación.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-11, RU-A-12.

Tabla 65: RF-A-15; Mostrar información app.

ID	RF-A-16
Título	Enviar correo al administrador.
Descripción	El sistema debe abrir la aplicación por defecto del sistema para los emails.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-11, RU-A-12.

Tabla 66: RF-A-16; Enviar correo al administrador.

ID	RF-A-17
Título	Formulario configurar modo TAG.
Descripción	El sistema debe mostrar un formulario para configurar el modo TAG.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-07, RU-A-09, RU-A-13, RU-A-14.

Tabla 67: RF-A-17; Formulario configurar modo TAG.

ID	RF-A-18
Título	Formulario configurar modo Reader/Verifier.
Descripción	El sistema debe mostrar un formulario para configurar el modo Reader/Verifier.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-07, RU-A-09, RU-A-15, RU-A-16.

Tabla 68: RF-A-18; Formulario configurar modo Reader/Verifier.

ID	RF-A-19
Título	Selección de TAGs/Grupos.
Descripción	El sistema debe mostrar los TAGs y Grupos ya existentes en la base de datos para su selección en forma de "*spinner*".
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-07, RU-A-09, RU-A-13, RU-A-14, RU-A-15, RU-A-16.

Tabla 69: RF-A-19; Selección de TAGs/Grupos.

ID	RF-A-20
Título	Permitir configurar el tiempo por prueba (proof) del protocolo.
Descripción	El sistema debe permitir, en el formulario de configuración del modo Reader/Verifier, configurar el tiempo por prueba (proof). Este campo será numérico con un rango: [2 – 99] segundos.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-15, RU-A-16.

Tabla 70: RF-A-20; Permitir configurar el tiempo por prueba (proof) del protocolo.

ID	RF-A-21
Título	Permitir conectar dispositivos Android o Arduino.
Descripción	El sistema debe permitir, en el formulario de configuración del modo Reader/Verifier, asignar si el TAG a conectar es Android o Arduino.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-15.

Tabla 71: RF-A-21; Permitir conectar dispositivos Android o Arduinos.

ID	RF-A-22
Título	Avisos en formularios.
Descripción	El sistema debe de avisar si el formulario está relleno correctamente o no.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-01, RU-A-05, RU-A-06, RU-A-08, RU-A-13, RU-A-15.

Tabla 72: RF-A-22; Avisos en formularios.

ID	RF-A-23
Título	Ejecutar/Parar modo TAG del protocolo Kazahaya.
Descripción	El sistema debe ejecutar el protocolo Kazahaya en modo TAG cuando el usuario presione un botón, y pararlo cuando presione otro.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-04, RU-A-13, RU-A-14.

Tabla 73: RF-A-23; Ejecutar/Parar modo TAG del protocolo Kazahaya.

ID	RF-A-24
Título	Ejecutar/Parar modo Reader/Verifier del protocolo Kazahaya.
Descripción	El sistema debe ejecutar el protocolo Kazahaya en modo Reader/Verifier cuando el usuario presione un botón, y pararlo cuando presione otro.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-04, RU-A-15, RU-A-16.

Tabla 74: RF-A-24; Ejecutar/Parar modo Reader/Verifier del protocolo Kazahaya.

ID	RF-A-25
Título	Aviso de problemas durante el protocolo Kazahaya.
Descripción	El sistema debe informar de problemas durante el protocolo detallando la causa del problema.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-14, RU-A-16.

Tabla 75: RF-A-25; Aviso de problemas durante el protocolo Kazahaya.

ID	RF-A-26
Título	Cifrar los sellados de tiempo
Descripción	El sistema debe cifrar los sellados de tiempo usados en el protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☐Alta ☒Media ☐Baja
Trazabilidad	RU-A-14, RU-A-16.

Tabla 76: RF-A-26; Cifrar los "timestamps".

ID	RF-A-27
Título	Guardar las tuplas del protocolo en una base de datos
Descripción	El sistema debe guardar todas las tuplas en la base de datos de la aplicación.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☐Alta ☒Media ☐Baja
Trazabilidad	RU-A-06, RU-A-08, RU-A-14, RU-A-16.

Tabla 77: RF-A-27; Guardar las tuplas del protocolo en una base de datos.

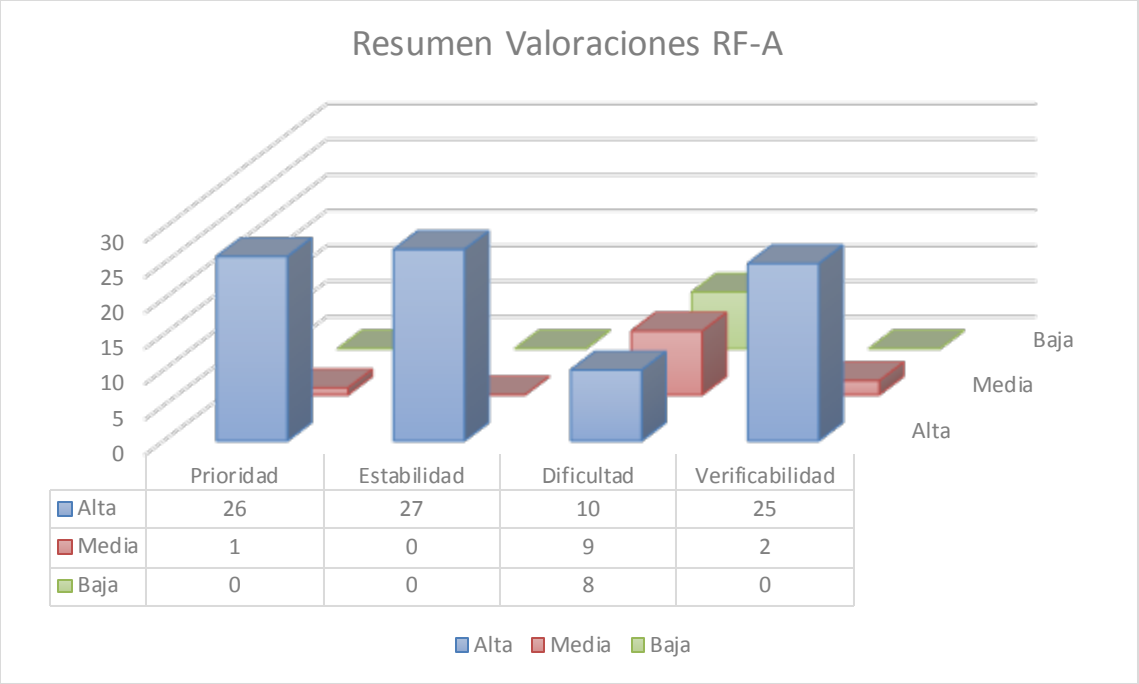


Figura 28: Resumen valoraciones requisitos funcionales aplicación Android.

RF-A/RU-A	RU-A-01	RU-A-02	RU-A-03	RU-A-04	RU-A-05	RU-A-06	RU-A-07	RU-A-08	RU-A-09	RU-A-10	RU-A-11	RU-A-12	RU-A-13	RU-A-14	RU-A-15	RU-A-16
RF-A-01	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RF-A-02	X														X	X
RF-A-03		X	X	X	X					X			X	X	X	X
RF-A-04		X	X	X	X											
RF-A-05		X	X	X	X										X	X
RF-A-06		X	X	X	X								X	X	X	X
RF-A-07						X							X	X	X	X
RF-A-08							X						X	X	X	X
RF-A-09								X					X	X	X	X
RF-A-10									X				X	X	X	X
RF-A-11						X		X					X	X	X	X
RF-A-12						X		X					X	X	X	X
RF-A-13						X		X					X	X	X	X
RF-A-14			X	X	X								X		X	
RF-A-15											X	X				
RF-A-16											X	X				
RF-A-17							X		X				X	X		
RF-A-18							X		X						X	X
RF-A-19							X		X				X	X	X	X
RF-A-20															X	X
RF-A-21															X	
RF-A-22	X				X	X		X					X		X	
RF-A-23				X									X	X		
RF-A-24				X											X	X
RF-A-25														X		X
RF-A-26														X		X
RF-A-27						X		X						X		X

Tabla 78: Matriz de Trazabilidad RF-A / RU-A.

3.5.5 Requisitos Funcionales programa Arduino UNO

En esta sección se van a definir los requisitos de software funcionales del sistema (programa Arduino UNO):

ID	RF-AU-01
Título	Cambio de PIN del dispositivo bluetooth.
Descripción	El sistema debe cambiar el PIN del dispositivo bluetooth cuando se requiera, siendo la longitud máxima de éste de 4.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-01, RU-AU-03, RU-AU-06.

Tabla 79: RF-AU-01; Cambio de PIN del dispositivo bluetooth.

ID	RF-AU-02
Título	Cambio de Nombre del dispositivo bluetooth.
Descripción	El sistema debe cambiar el Nombre del dispositivo bluetooth cuando se requiera, siendo la longitud máxima de éste de 20 caracteres.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-02, RU-AU-03, RU-AU-06.

Tabla 80: RF-AU-02; Cambio de Nombre del dispositivo bluetooth.

ID	RF-AU-03
Título	Establecer conexión.
Descripción	El sistema debe tener el dispositivo listo para una conexión.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-03, RU-AU-06.

Tabla 81: RF-AU-03; Establecer conexión.

ID	RF-AU-04
Título	Identificadores de TAG y de Grupo.
Descripción	El sistema debe permitir el cambio de los identificadores de TAG y de Grupo numéricos, siendo la longitud máxima de éste de 9
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-04, RU-AU-06.

Tabla 82: RF-AU-04; Identificadores de TAG y de Grupo.

ID	RF-AU-05
Título	Claves de TAG y de Grupo.
Descripción	El sistema debe permitir el cambio de las Claves de TAG y de Grupo numéricas, siendo la longitud máxima de ésta de 9.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-03, RU-AU-06.

Tabla 83: RF-AU-05; Claves de TAG y de Grupo.

ID	RF-AU-06
Título	Ejecutar el modo TAG.
Descripción	El sistema debe ejecutar el modo TAG y esperar a que alguien se empareje y conecte para comenzar el protocolo.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-AU-XX.

Tabla 84: RF-AU-06; Claves de TAG y de Grupo.

ID	RF-AU-07
Título	Aviso de problemas durante el modo TAG.
Descripción	El sistema debe avisar mediante un altavoz y un led de problemas durante la ejecución del modo TAG del protocolo Kazahaya.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-06.

Tabla 85: RF-AU-07; Aviso de problemas durante el modo TAG.

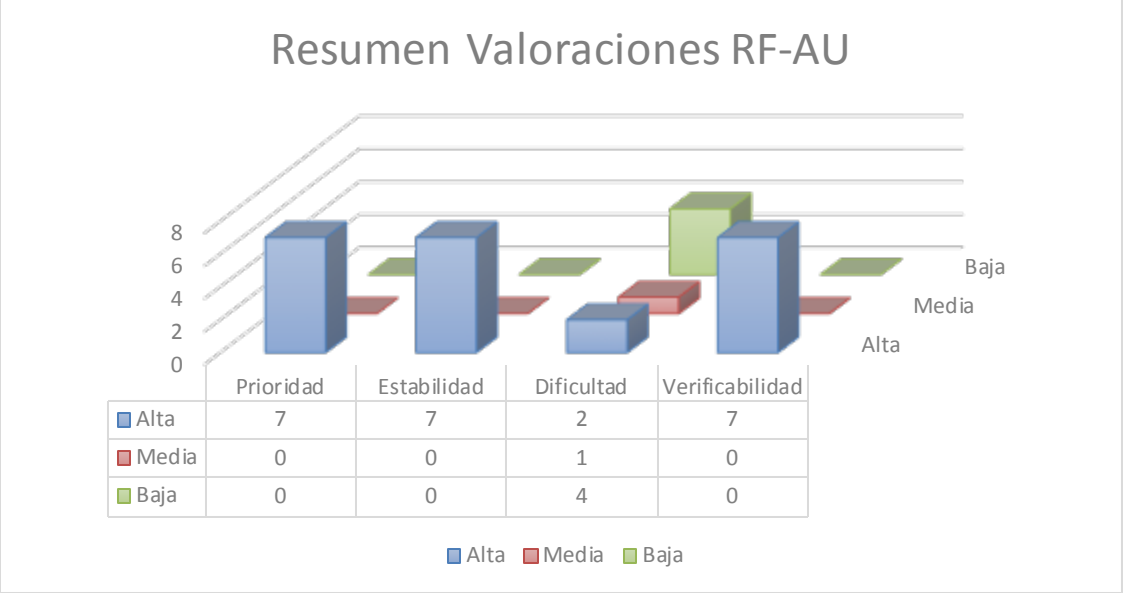


Figura 29: Resumen valoraciones requisitos funcionales programa Arduino UNO.

RF-AU/RU-AU	RU-AU-01	RU-AU-02	RU-AU-03	RU-AU-04	RU-AU-05	RU-AU-06
RF-AU-01	x		x			x
RF-AU-02		x	x			x
RF-AU-03			X			X
RF-AU-04				X		X
RF-AU-05			X			X
RF-AU-06	X	X	X	X	X	X
RF-AU-07						X

Tabla 86: Matriz de Trazabilidad RF-AU / RU-AU.

3.5.6 Requisitos NO Funcionales aplicación Android

En esta sección se van a definir los requisitos de software NO funcionales del sistema (aplicación Android):

ID	RNF-A-01
Título	Comunicaciones cifradas por bluetooth.
Descripción	El sistema debe usar el canal cifrado que proporciona la API bluetooth de Android.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☒Alta ☐Media ☐Baja
Verificabilidad	☐Alta ☒Media ☐Baja
Trazabilidad	RU-A-14, RU-A-16.

Tabla 87: RNF-A-01; Comunicaciones cifradas por bluetooth.

ID	RNF-A-02
Título	Ejecución del protocolo en segundo plano.
Descripción	El sistema debe poder ejecutar el protocolo en segundo plano.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-14, RU-A-16.

Tabla 88: RNF-A-02; Ejecución del protocolo en segundo plano.

ID	RNF-A-03
Título	Interfaz usable y corporativa.
Descripción	El sistema debe tener una interfaz usable y a la vez corporativa. A su vez la API Android a usar será la 16, para permitir el diseño usable aplicado.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-A-XX.

Tabla 89: RNF-A-03; Interfaz usable y corporativa.

ID	RNF-A-04
Título	Implementación Java.
Descripción	El sistema debe estar implementado en Java para correr en dispositivos Android.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-A-XX.

Tabla 90: RNF-A-04; Implementación Java.

ID	RNF-A-05
Título	Tiempo de ejecución del protocolo.
Descripción	El sistema debe ejecutar una pasada del protocolo en menos de 2 segundos.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-A-14, RU-A-16.

Tabla 91: RNF-A-05; Tiempo de ejecución del protocolo.

ID	RNF-A-06
Título	Modulado y orientado a Objetos.
Descripción	El sistema debe de seguir un aspecto modulado y orientado a objetos para su buen mantenimiento/entendimiento.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-A-XX.

Tabla 92: RNF-A-06; Modulado.

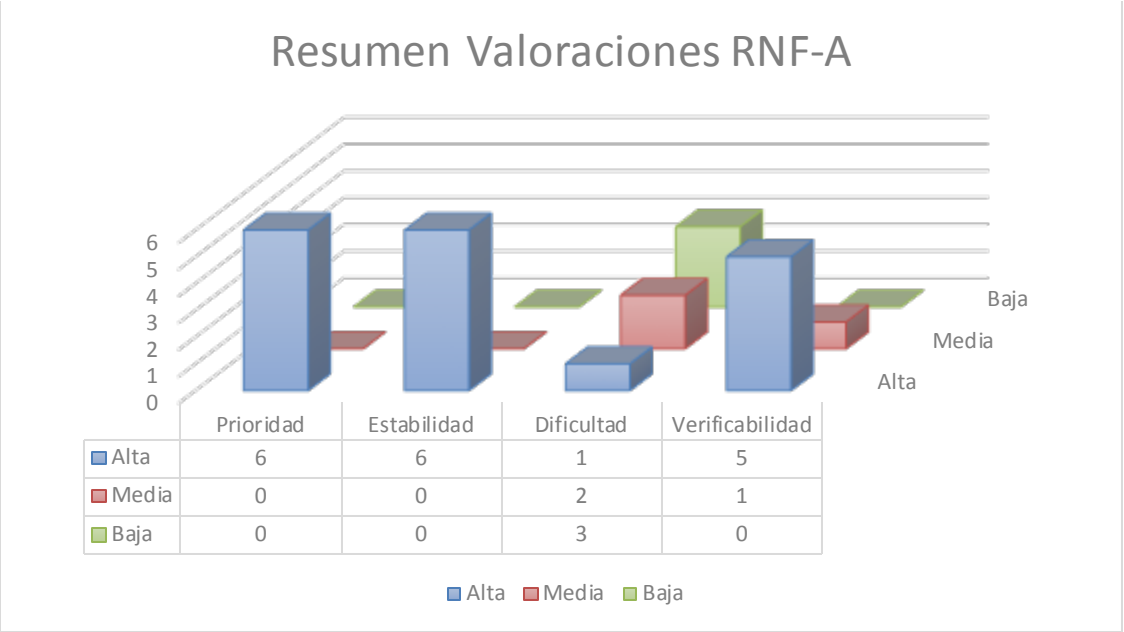


Figura 30: Resumen valoraciones requisitos NO funcionales aplicación Android.

RNF-A/RU-A	RU-A-01	RU-A-02	RU-A-03	RU-A-04	RU-A-05	RU-A-06	RU-A-07	RU-A-08	RU-A-09	RU-A-10	RU-A-11	RU-A-12	RU-A-13	RU-A-14	RU-A-15	RU-A-16
RNF-A-01														X		X
RNF-A-02														X		X
RNF-A-03	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RNF-A-04	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RNF-A-05														X		X
RNF-A-06	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabla 93: Matriz de Trazabilidad RNF-A / RU-A.

3.5.7 Requisitos NO funcionales programa Arduino UNO

En esta sección se van a definir los requisitos de software NO funcionales del sistema (programa Arduino UNO):

ID	RNF-AU-01
Título	Implementación en C.
Descripción	El sistema debe estar implementado en C para ser ejecutado en la plataforma Arduino UNO.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-AU-XX.

Tabla 94: RNF-AU-01; Implementación en C.

ID	RNF-AU-02
Título	Tiempo de ejecución del protocolo.
Descripción	El sistema debe ejecutar una pasada del protocolo en menos de 1 segundos.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☒Media ☐Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	RU-AU-06.

Tabla 95: RNF-AU-02; Tiempo de ejecución del protocolo.

ID	RNF-AU-03
Título	Modulado.
Descripción	El sistema debe de seguir un aspecto modulado de las funciones y métodos usados.
Prioridad	☒Alta ☐Media ☐Baja
Estabilidad	☒Alta ☐Media ☐Baja
Dificultad	☐Alta ☐Media ☒Baja
Verificabilidad	☒Alta ☐Media ☐Baja
Trazabilidad	Todos los RU-A-XX.

Tabla 96: RNF-AU-03; Modulado.

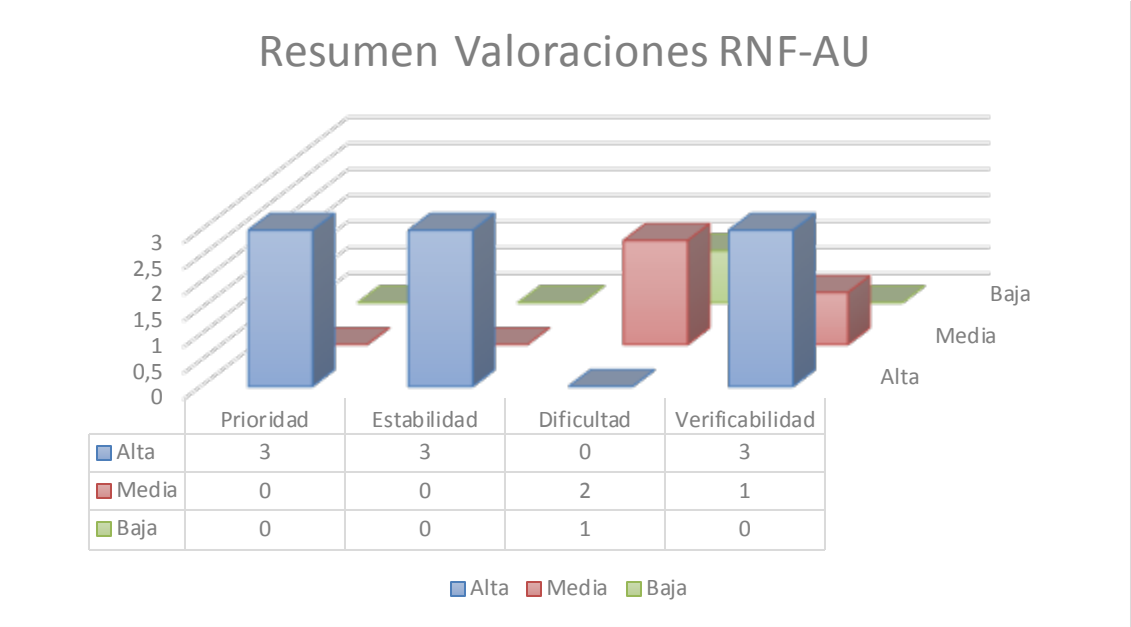


Figura 31: Resumen valoraciones requisitos NO funcionales programa Arduino UNO.

RNF-AU/RU-AU	RU-AU-01	RU-AU-02	RU-AU-03	RU-AU-04	RU-AU-05	RU-AU-06
RNF-AU-01	X	X	X	X	X	X
RNF-AU-02						X
RNF-AU-03	X	X	X	X	X	X

Tabla 97: Matriz de Trazabilidad RNF-AU / RU-AU.

CAPÍTULO 4

DISEÑO

Después de haber realizado el Análisis del Sistema, en este capítulo vamos a adentrarnos con el Diseño del mismo, donde detallaremos la arquitectura del sistema, diagrama de clases, modelo de datos y diseño de las ventanas de la aplicación Android.

4.1 ARQUITECTURA DEL SISTEMA

Nuestro sistema completo y funcional está formado por dos sistemas, implementados en dispositivos Android y Arduino respectivamente. que se complementan el uno al otro para la correcta ejecución del protocolo estudiado. A continuación se va a explicar cada uno de ellos:

4.1.1 Arquitectura

Android

El patrón seguido para la arquitectura de software de la aplicación Android es el conocido como Modelo-Vista-Controlador (MVC). Éste modelo se basa fundamentalmente en la división por componentes:

- **Modelo:** se encarga del acceso a los datos que necesita la aplicación.
- **Vista:** es la interfaz con la cual el usuario interactúa con el sistema.
- **Controlador:** se encarga de toda la lógica de control del sistema y es la que interactúa con las otras dos partes anteriormente mencionadas.

Figura 32: Modelo-Vista-Controlador (MVC Android).

Arduino UNO

Al tratarse de un programa escrito en C para la plataforma Arduino, la arquitectura que hemos seguido para el mismo ha sido basada en la programación modular. Por ello, sólo consta de un fichero con un listado de operaciones colocadas en orden de ejecución y tratadas mediante funciones para mantener una buena estructura, lo que facilita las labores de depuración y reutilización de código.

4.1.2 Diagramas

En esta sección se va a presentar una visión de los paquetes y clases que componen nuestra aplicación Android, la cual puede ejercer de Reader/Verifier o de TAG, y otra del programa Arduino UNO. Ambos en conjunto forman nuestro sistema completo.

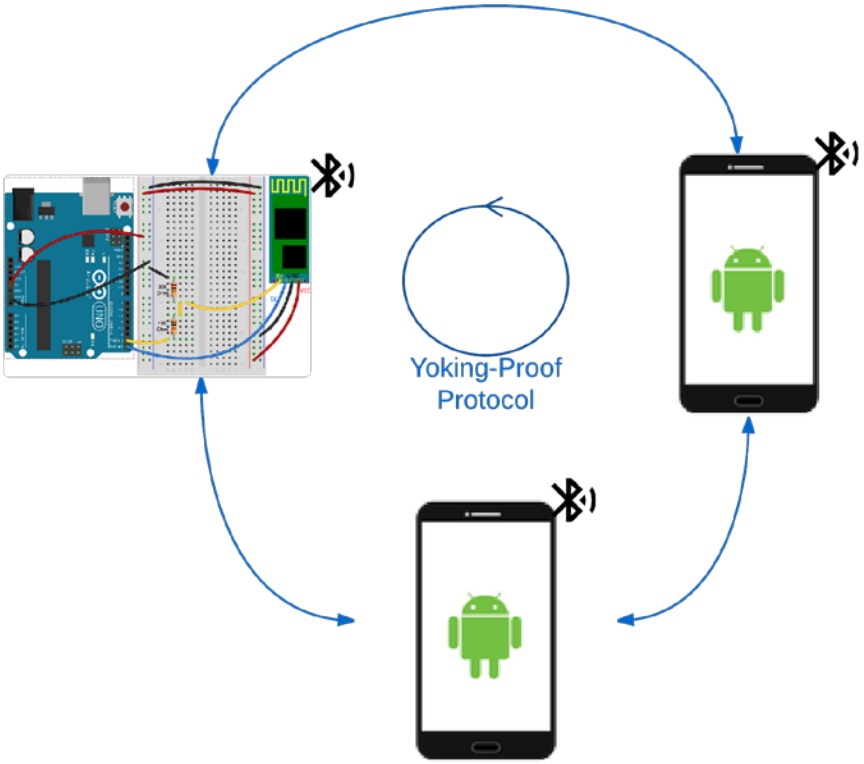


Figura 33: Diagrama del Sistema completo.

Diagrama de Clases aplicación Android

La aplicación Android, con arquitectura MVC, está compuesta de varios paquetes y dentro de ellos se encuentran las clases que hemos creado. En la Figura 34 se muestra el diagrama de clases:

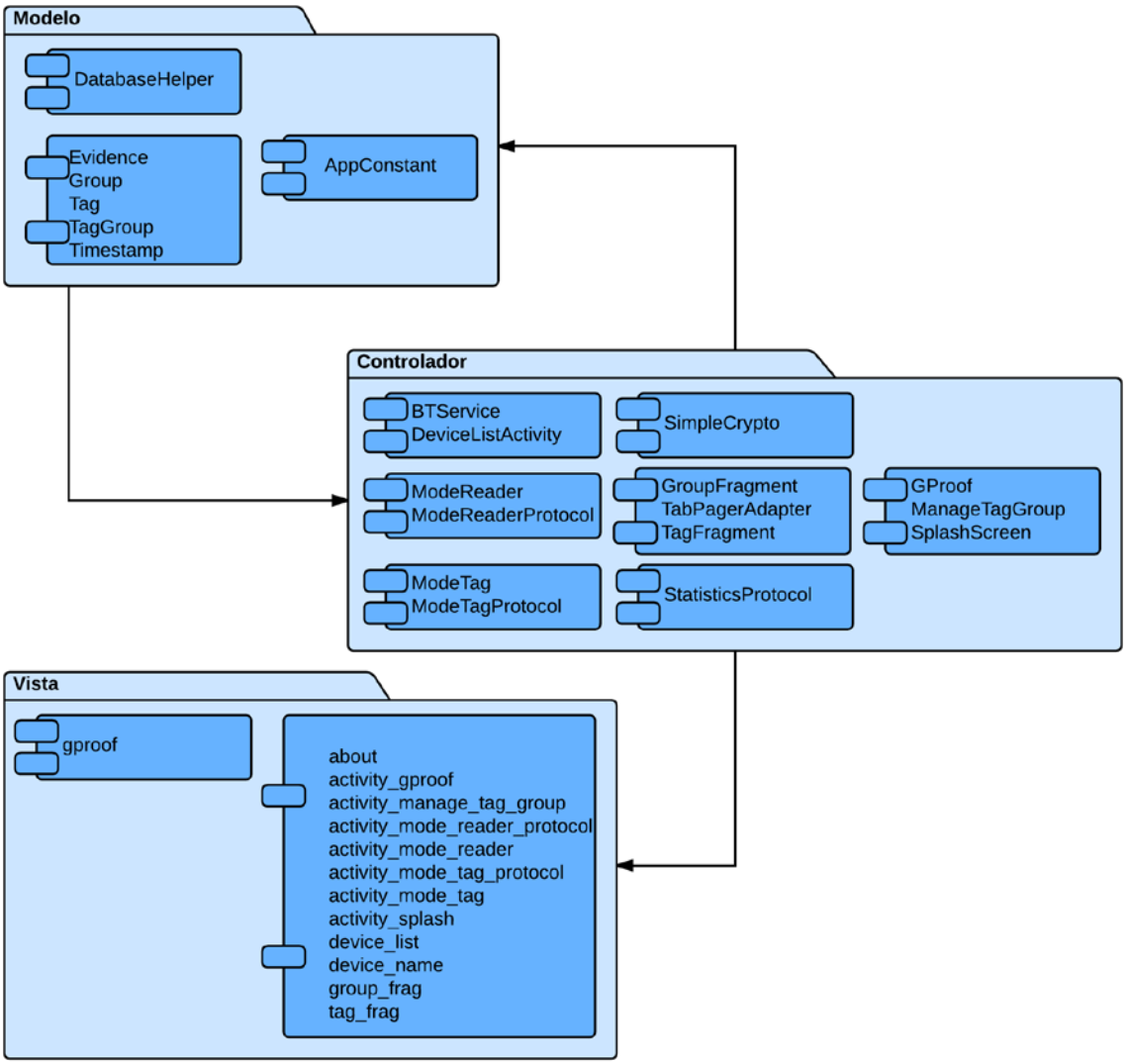


Figura 34: Diagrama de Clases aplicación Android.

Como se puede observar en la figura, se presenta un diagrama de clases en relación a la arquitectura usada (MVC). Dentro de cada subsistema podemos ver los paquetes que le forman, y dentro de éstos las clases correspondientes.

Las clases que forman la Vista se tratan de ficheros XML, y las clases que forman el Modelo y el Controlador son ficheros Java.

Para consultar la gran cantidad de métodos y variables globales que forman cada una de estas clases, se puede acceder al *Javadoc* generado en la aplicación.

Diagrama del programa Arduino UNO

El programa Arduino UNO, al estar escrito en C, presenta un modelo estructurado en un único fichero con la extensión de esta plataforma *"INO"*:

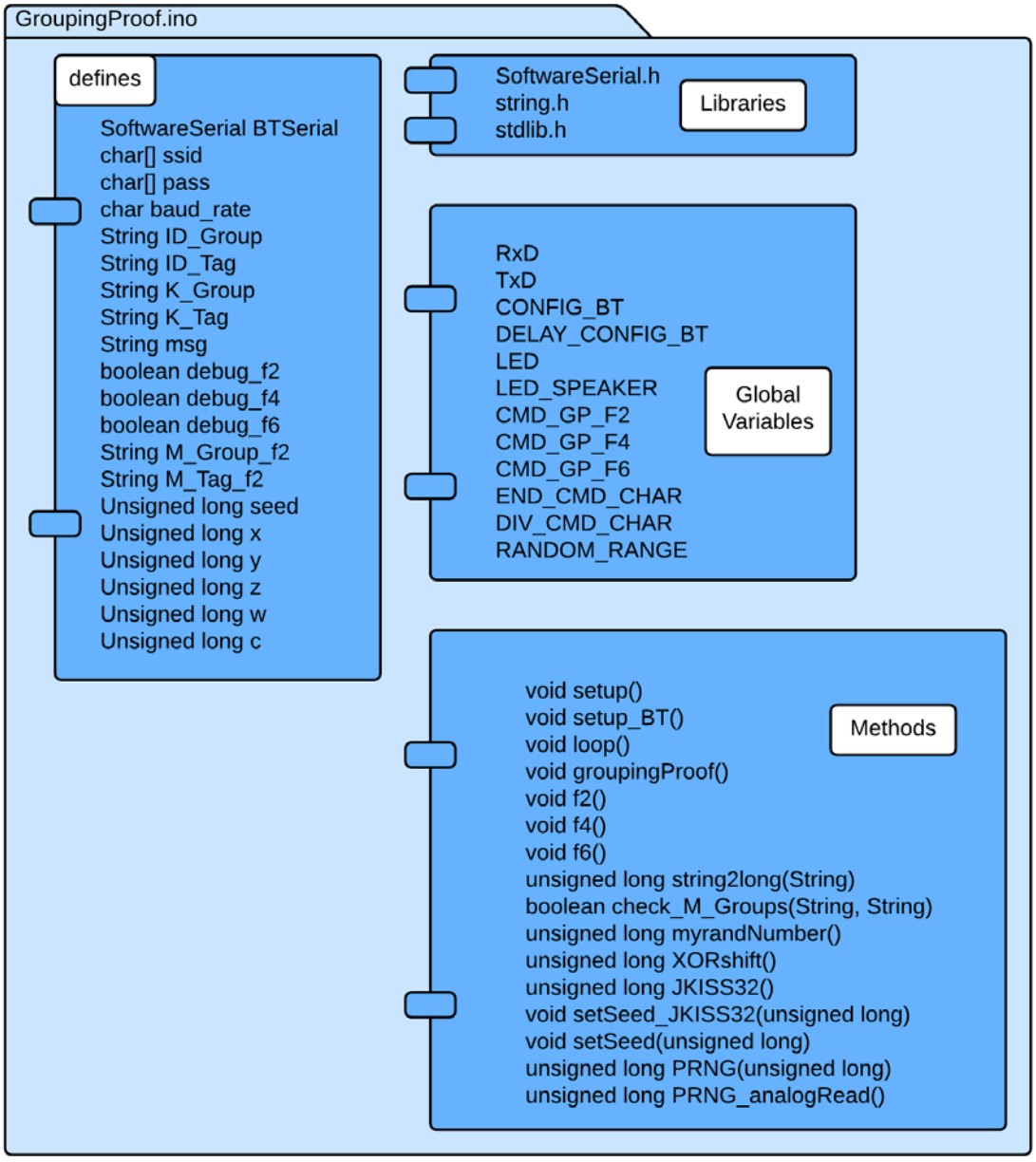


Figura 35: Diagrama modelo estructurado programa Arduino UNO.

Como se puede observar en la Figura 35, el modelo de Arduino se ha dividido en partes para estructurarlo en distintas partes o módulos y así proporcionar un diseño más visual. Estas partes son las librerías, las constantes, las variables globales y los métodos.

4.1.3 Circuito electrónico Arduino UNO

Para el diseño de la parte del modo TAG del protocolo Kazahaya en la plataforma Arduino UNO, se ha hecho uso de un circuito electrónico que incluye el dispositivo bluetooth, y un altavoz y un LED para el sistema de alarma.

Los componentes utilizados han sido los siguientes:

- Microcontrolador Arduino UNO.
- 1 placa de prototipado.
- 1 LED (Light-Emitting Diode).
- 1 resistencia de 220 ohmios.
- 1 dispositivo bluetooth HC-06.
- 1 altavoz.
- Cable rígido para la interconexión de los componentes.

Las conexiones que hay que realizar son:

- **Altavoz:** el positivo se conecta al pin 12 del Arduino y el negativo al pin GND.
- **Bluetooth:** este dispositivo tiene 4 pines:
  - VCC se conecta al pin de 5V del Arduino.
  - GND se conecta al pin GND del Arduino.
  - TXD se conecta al pin configurado como RXD en el Arduino (10).
  - RXD se conecta al pin configurado como TXD en el Arduino (11).
- **LED:** la patilla corta (negativo) se conecta a una resistencia de 220 ohmios y ésta a su vez al GND del Arduino. La patilla larga (positivo) se conecta al pin 13 del Arduino.

A continuación se muestra el prototipado y el esquema del circuito con los componentes y las conexiones:

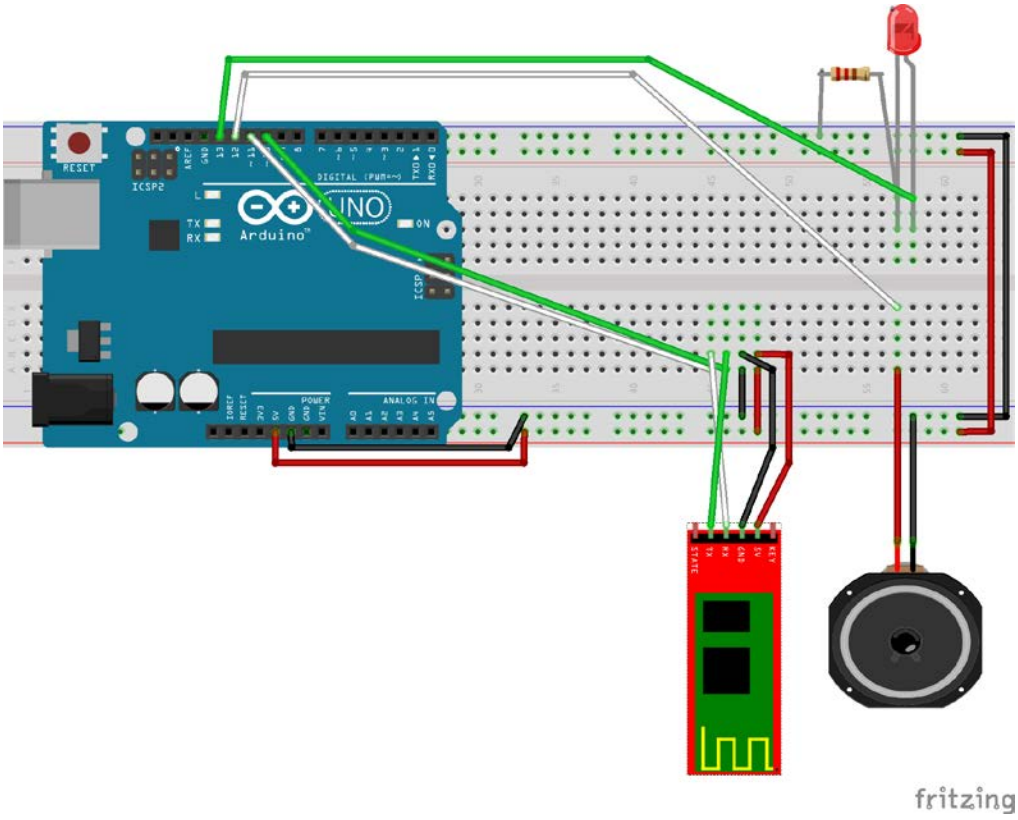


Figura 36: Prototipado sistema Arduino UNO.

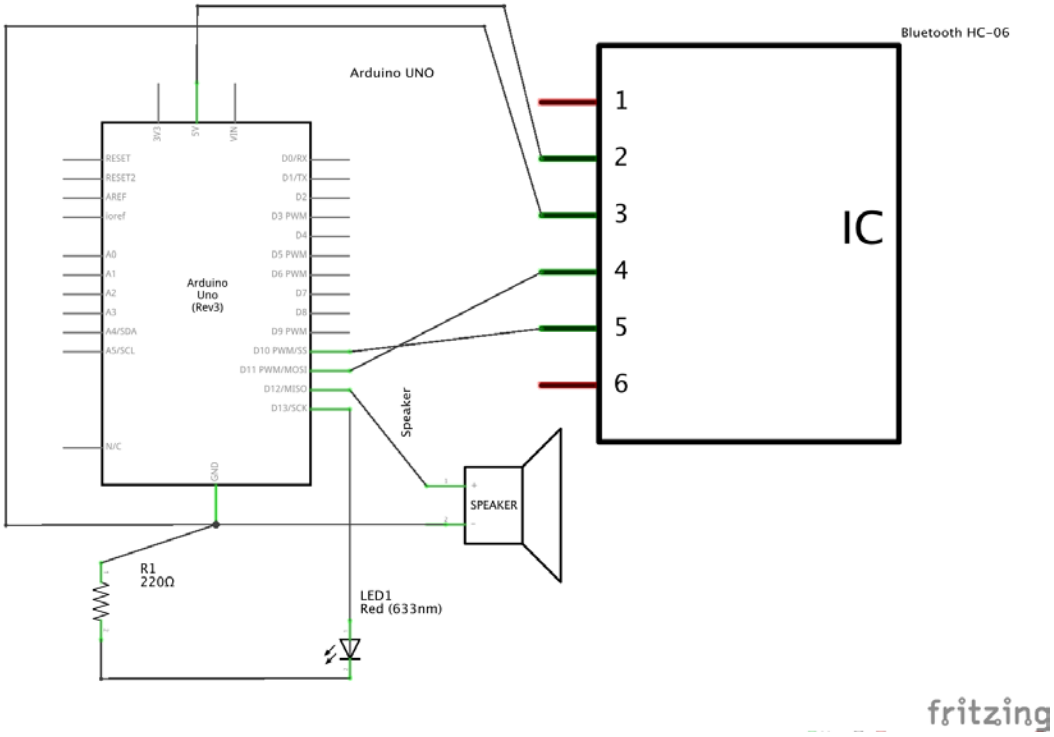


Figura 37: Esquema sistema Arduino UNO.

4.1.4 Flujo protocolo Kazahaya

Para su correcto funcionamiento y con respecto a la explicación del protocolo *(véase 3.3.2)*, se ha creado un flujo de control gestionado mediante llamadas a las distintas funciones principales que componen el mismo y el modelo de datos *(véase 4.1.5)*.

Los pasos principales de una pasada en el protocolo son:

1. El Lector/Verificador ejecuta la función **f1()** enviando el *timestamp*.
2. El TAG A recibe los datos enviados por la función f1() y son derivados a la función **f2()** para su procesado y envío de los nuevos datos.
3. Posteriormente el Lector/Verificador recibe estos datos para su procesado en la función **f3()** y envío de los nuevos datos al TAG B.
4. El TAG B recibe estos datos y pasa a procesarlos a la función **f4()**, la cual vuelve a enviar los nuevos datos en el caso de que no haya ninguna alerta.
5. El Lector/Verificador recibe los datos y pasa a procesarlos a la función **f5()**, enviando los nuevos datos al TAG A de nuevo.
6. El TAG A realiza una serie de cálculos en la función **f6()** y reenvía los nuevos datos de vuelta.
7. Finalmente, el Lector/Verificador se encarga de generar la Evidencia en la función **f7()**.

A continuación se muestra gráficamente este flujo para que se pueda seguir de una manera visual:

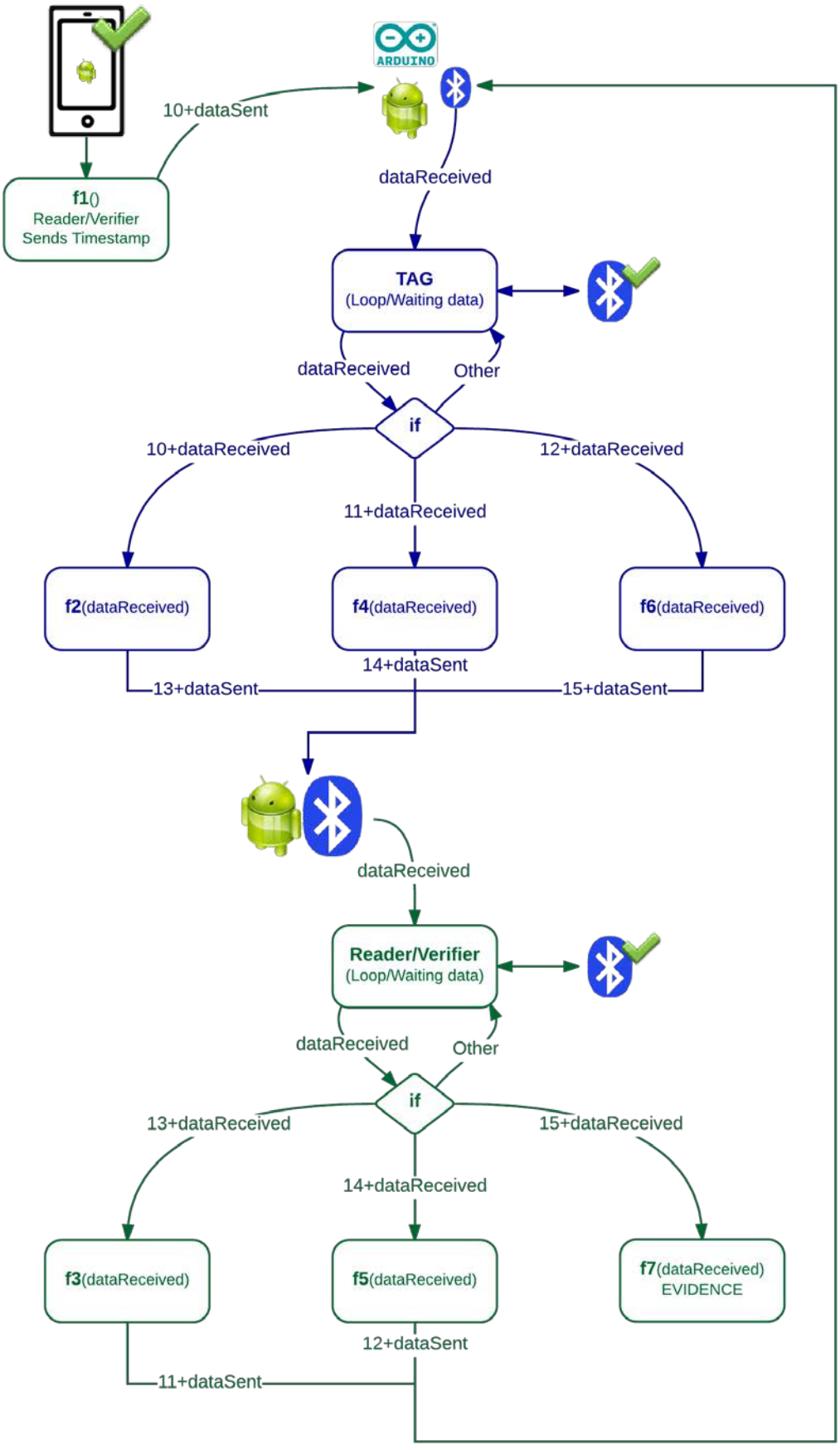


Figura 38: Flujo de ejecución del protocolo Kazahaya.

4.1.5 Modelo de Datos

En este apartado vamos a presentar el diseño de la base de datos desarrollado para la aplicación Android y el diseño creado para el paso de mensajes del protocolo Kazahaya.

Base de datos Android

Debido a los requisitos estudiados en el análisis, es necesario el uso de un sistema de almacenamiento para poder manejar los datos que requiere el protocolo Kazahaya.

Para ello se ha usado las funcionales que proporciona Android mediante SQLite, permitiendo hacer operaciones sobre datos de una manera rápida y sencilla. El diseño de la base de datos es el siguiente:

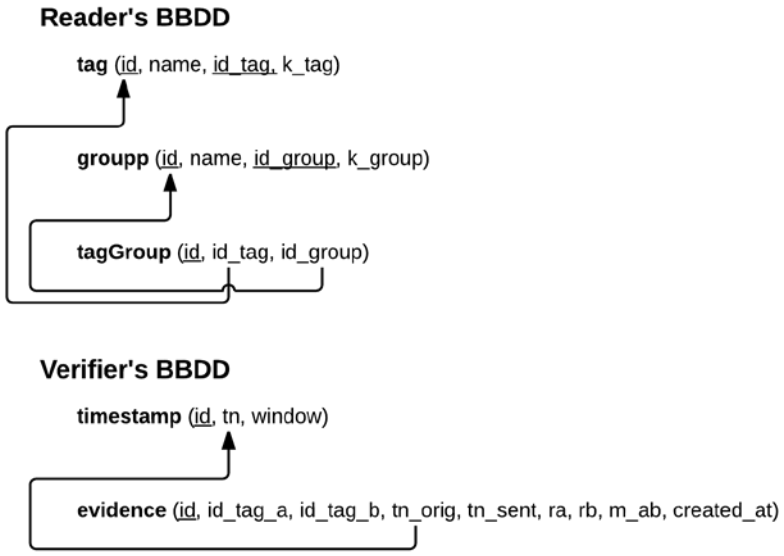


Figura 39: Diseño de la base de datos de la aplicación Android.

Para ello se ha divido la base de datos virtualmente en dos partes, BBDD Lector y BBDD Verificador, aunque físicamente estén dentro del mismo dispositivo en nuestro desarrollo.

- **Tabla "tag":** el *id* es clave primaria e *integer*, los demás campos son *varchar*, además de que el campo *id_tag* es clave secundaria, previniendo de tal manera identificadores de etiqueta (tags) duplicados.
- **Tabla "group":** el *id* es clave primaria e *integer*, los demás campos son *varchar*, además de que el campo *id_group* es clave secundaria, previniendo de tal manera identificadores de grupo (groups) duplicados.

- **Tabla "tagGroup":** todos los campos son *integer*, el *id* es clave primaria, los demás campos son identificadores con clave foránea o *foreign key* a otras tablas, manteniendo de esta forma la integridad de los datos.
- **Tabla "timestamp":** el *id* es clave primaria e *integer*, los demás campos son *varchar*.
- **Tabla "evidence":** el *id* es clave primaria e *integer*, los demás campos son *varchar* formados con los datos de las demás tablas, y con una clave foránea o *foreign key* a la tabla *timestamp* para la integridad de los mismos.

Paso de mensajes protocolo Kazahaya

Para saber cómo tratar los datos enviados y recibidos por bluetooth durante el funcionamiento del protocolo, se ha diseñado el siguiente formato para paso de mensajes:

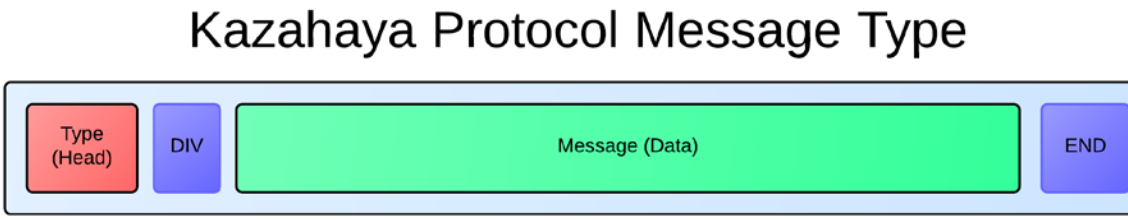


Figura 40: Paso de Mensajes protocolo Kazahaya.

Como se puede observar en la Figura 40, el mensaje consta de varias partes:

- **Tipo:** su función es definir a qué función va dirigido el mensaje.

Tipo	Función
10	F2()
11	F4()
12	F6()
13	F3()

14	F5()
15	F7()

Tabla 98: Correlación "Tipo – Función", Paso de mensajes protocolo.

- **DIV:** es un carácter (|) que se usa para dividir las partes del mensaje, así como todo lo interno que vaya en la parte de los datos (data).
- **Mensaje:** contiene todos los datos a tratar en el protocolo, dividiéndolos con DIV entre los diferentes tipos que pueda haber.
- **END:** es un carácter (#) que se usa para indicar el final del mensaje.

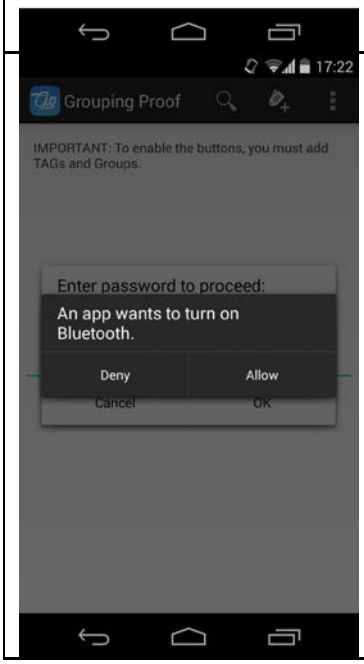
4.1.6 Interfaz de usuario

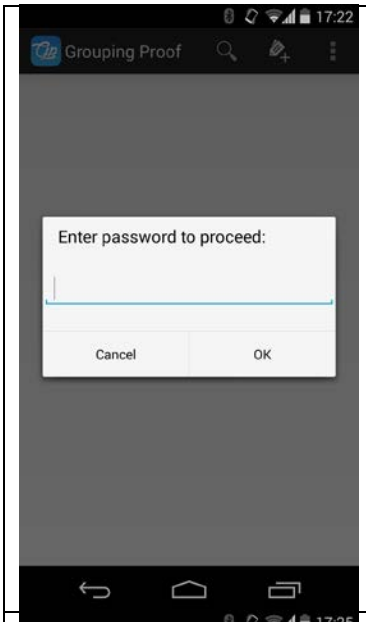
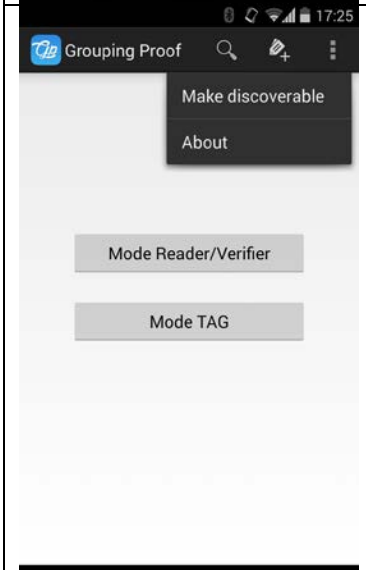
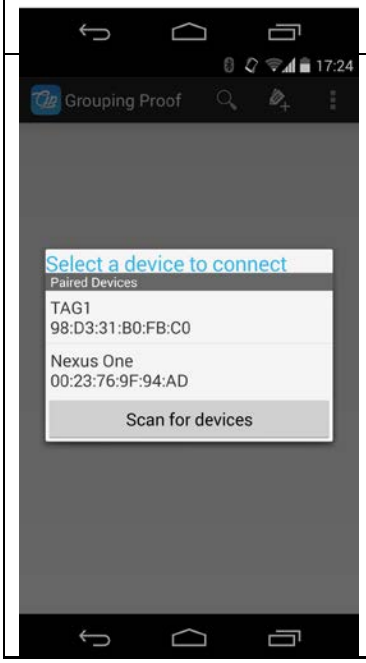
La interfaz de usuario se corresponde con distintas vistas de la aplicación desarrollada en Android. En este punto primero se va a presentar el flujo de ventanas de la aplicación Android, y posteriormente se detalla cada una de ellas proporcionando una breve descripción.

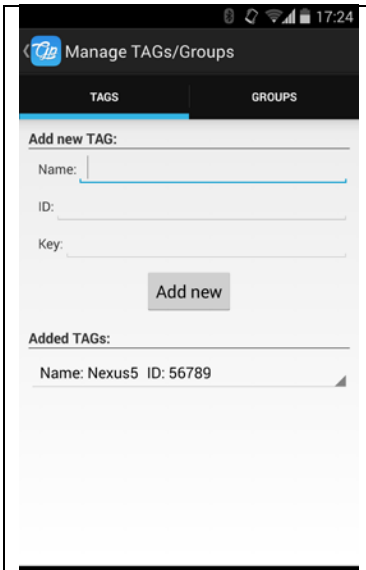
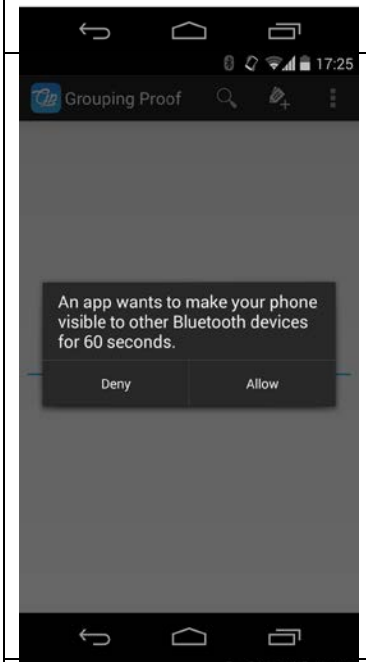
Flujo de ventanas aplicación Android

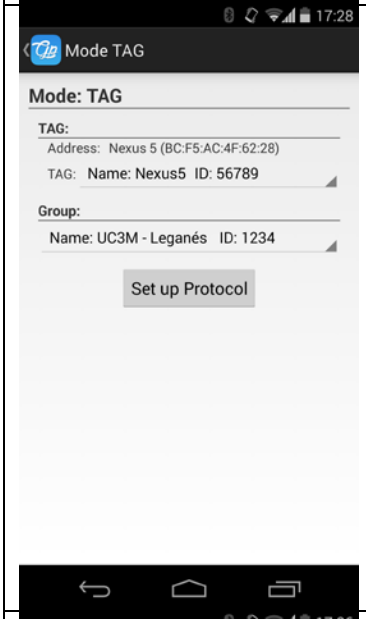
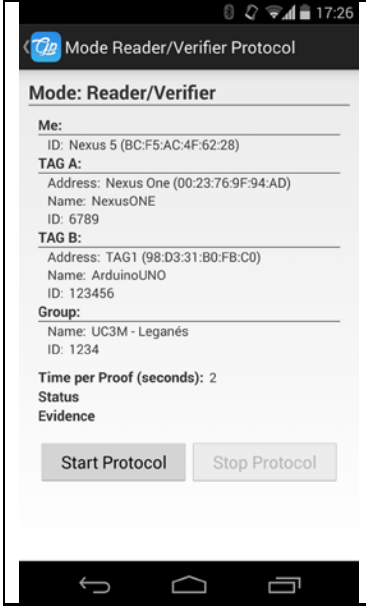
Figura 41: Flujo de ventanas aplicación Android.

Descripción de las ventanas

Ventana	Descripción
	Muestra una ventana de inicio de la aplicación.
	Pregunta por Activar o Desactivar el bluetooth.

	Pregunta por una clave para poder cifrar el *timestamp*.
	Es la pantalla principal de la aplicación, en la que se ofrece configurar cada uno de los modos disponibles, y una sección de menú, donde puedes ir a las ventanas de emparejar dispositivos, administrar TAGs/Grupos, hacer el dispositivo visible o Acerca de la aplicación.
	Nos muestra el diálogo para ver los dispositivos emparejados, buscar dispositivos y emparejar nuevos dispositivos.

	Esta ventana nos muestra la administración de los TAGs o Grupos, donde podemos añadir nuevos o consultar los ya existentes.
	Tras pulsar en el menú el botón "Hacer visible", nos muestra el diálogo para hacer visible por bluetooth durante 1 minuto nuestro dispositivo.
	Tras pulsar en el menú el botón "Acerca de", nos muestra un diálogo con la información de la aplicación.

	Tras pulsar en el botón "Modo Reader/Verifier", nos muestra un formulario de configuración de dicho modo.
	Tras pulsar en el botón "Modo TAG", nos muestra un formulario de configuración de dicho modo.
	Tras pulsar en el botón "Configurar Protocolo" en el modo Reader/Verifier, nos muestra la ventana donde podemos empezar o parar el protocolo.

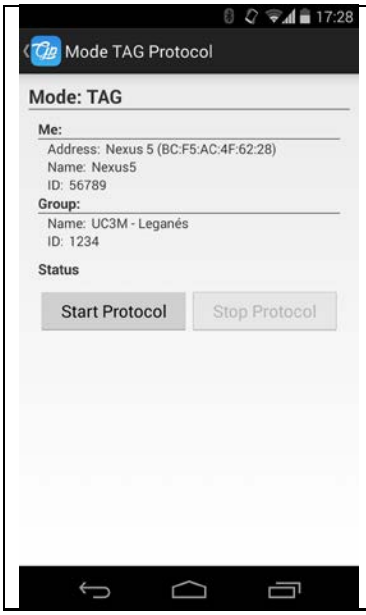
	Tras pulsar en el botón "Configurar Protocolo" en el modo TAG, nos muestra la ventana donde podemos empezar o parar el protocolo.

Tabla 99: Descripción de ventanas aplicación Android.

CAPÍTULO 5

IMPLEMENTACIÓN

Después de haber realizado el Diseño del Sistema, este capítulo explica detalles acerca de la implementación de las aplicaciones de Android y Arduino que componen el sistema, indicando las decisiones más relevantes que se han tomado.

5.1 APLICACIÓN ANDROID – YOKING PROOF.APK

La implementación de la aplicación "*Yoking Proof*" se ha realizado en Java mediante el SDK de Android proporcionado por Google. Se ha utilizado como base la API 16, y como máximo la API 19, por lo que la aplicación es compatible desde la versión Jelly Bean (v4.1.x) hasta la KitKat (4.4.4), abarcando de tal forma más del 75% del mercado *(véase 3.2.2)*.

Elementos principales utilizados

Para el desarrollo de la app se ha tenido muy en cuenta, ya que es muy importante, el ciclo de vida de las actividades (*activity)* en las aplicaciones Android. Es por ello que a continuación se presenta este ciclo:

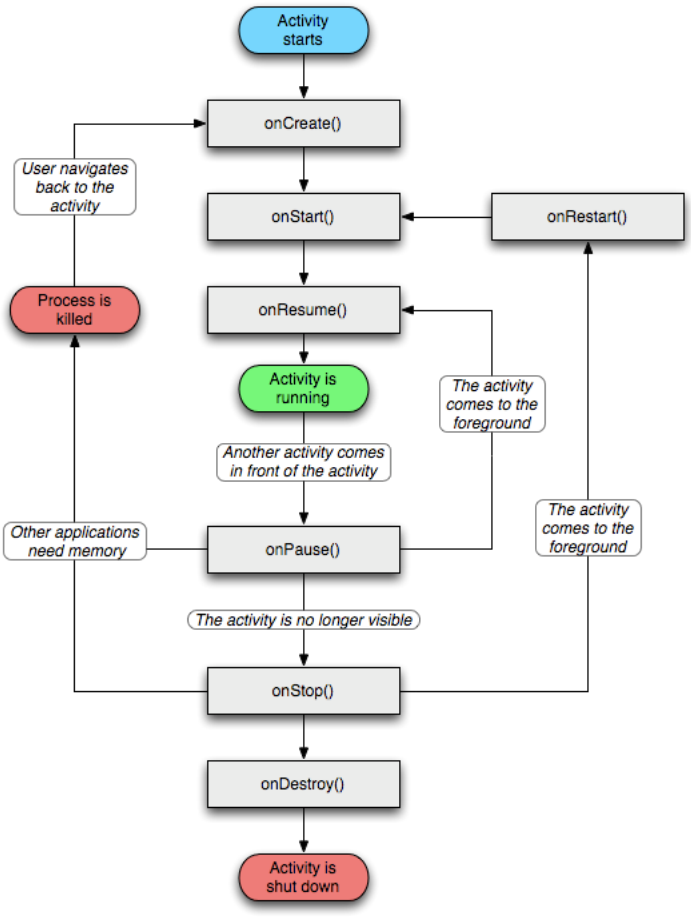


Figura 42: Ciclo de vida de una Actividad Android.

- **onCreate():** este método es llamado cuando la actividad se crea. Se usa para inicializar todo lo que se necesite como estructuras de datos o vistas.
- **onStart():** este método se llama cuando la actividad se hace visible al usuario.
- **onResume():** este método se llama cuando la actividad comienza a interactuar con el usuario.
- **onPause():** este método se lanza cuando la actividad está a punto de ser lanzada a segundo plano.
- **onStop():** este método se lanza cuando la actividad ya no es visible al usuario pero sigue en ejecución.
- **onRestart():** este método se lanza cuando la actividad va a volver a ser presentada al usuario después de haber pasado por onStop().
- **onDestroy():** este método es llamado cuando una actividad va a ser destruida por completo.

Otro de los elementos importantes usados en esta implementación es el uso de los **hilos (Threads)** y de los **manejadores (Handlers)** para el procesado en segundo plano de operaciones que requieran un largo tiempo de ejecución como es el caso de las conexiones bluetooth y de la implementación del protocolo Kazahaya.

El hilo es la función encargada de ejecutar tareas en segundo plano, pero la cual no puede insertar datos en el hilo principal de la aplicación (UI Thread – "User Interface thread") que es el cual usa el usuario para interactuar con la aplicación. Para poder hacer esto sin que se produzcan errores o cuelgues de la aplicación se usan los manejadores. A continuación se presenta una imagen que muestra perfectamente lo que se ha explicado:

Figura 43: Threads & Hlandlers Android.

Paquete "bluetooth_service"

Este paquete contiene dos clases:

- **BTservice.java:** la cual nos proporciona funcionalidad completa para crear y manejar conexiones con otros dispositivos bluetooh. Tiene tres métodos principales:
  - *AcceptThread:* crea un hilo para escuchar conexiones entrantes.

```
public void run() {
    if (D) Log.d(TAG, "Socket Type: " + mSocketType + " BEGIN mAcceptThread" + this);
    setName("AcceptThread" + mSocketType);

    BluetoothSocket socket = null;

    // Listen to the server socket if we're not connected
    while (mState != STATE_CONNECTED) {
        try {
            // This is a blocking call and will only return on a
            // successful connection or an exception
            socket = mmServerSocket.accept();
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + " accept() failed", e);
            break;
        }

        // If a connection was accepted
        if (socket != null) {
            synchronized (BTService.this) {
                switch (mState) {
                case STATE_LISTEN:
                case STATE_CONNECTING:
                    // Situation normal. Start the connected thread.
                    connected(socket, socket.getRemoteDevice(), mSocketType);
                    break;
                case STATE_NONE:
                case STATE_CONNECTED:
                    // Either not ready or already connected. Terminate new socket.
                    try {
                        socket.close();
                    } catch (IOException e) {
                        Log.e(TAG, "Could not close unwanted socket", e);
                    }
                    break;
                }
            }
        }
    }
```

Figura 44: Hilo para aceptar conexiones entrantes.

  - *ConnectThread:* crea un hilo para conectar con un dispositivo.

```
public void run() {
    Log.i(TAG, "BEGIN mConnectThread SocketType: " + mSocketType);
    setName("ConnectThread " + mSocketType);

    // Always cancel discovery because it will slow down a connection
    mAdapter.cancelDiscovery();

    // Make a connection to the BluetoothSocket
    try {
        // This is a blocking call and will only return on a
        // successful connection or an exception
        mmSocket.connect();
    } catch (IOException e) {
        // Close the socket
        try {
            mmSocket.close();
        } catch (IOException e2) {
            Log.e(TAG, "unable to close() " + mSocketType + " socket during connection failure", e2);
        }
        connectionFailed();
        return;
    }

    // Reset the ConnectThread because we're done
    synchronized (BTService.this) {
        mConnectThread = null;
    }

    // Start the connected thread
    connected(mmSocket, mmDevice, mSocketType);
}
```

Figura 45: Hilo para conectar dispositivos bluetooth.

- *ConnectedThread:* crea un hilo para realizar transmisiones de datos.

```
public void run() {
    Log.i(TAG, "BEGIN mConnectedThread");
    byte[] readBuffer = new byte [2048];
    int readBufferPosition = 0;
    String data = "";
    stopThread = false;

    // Keep listening to the InputStream while connected
    while (!stopThread) {
        try {
            data = "";
            int bytesAvailable = mmInStream.available();
            if(bytesAvailable > 0) {
                byte[] packetBytes = new byte[bytesAvailable];
                mmInStream.read(packetBytes);
                for(int i=0 ; i < bytesAvailable ; i++)
                {
                    byte b = packetBytes[i];
                    if(b == AppConstant.END_CMD_CHAR_BYTE)
                    {
                        byte[] encodedBytes = new byte[readBufferPosition];
                        System.arraycopy(readBuffer, 0, encodedBytes, 0, encodedBytes.length);
                        data = new String(encodedBytes, "US-ASCII");
                        readBufferPosition = 0;
                        Log.e(TAG, "STRING IN InputStream : " + data);
                    }
                    else
                    {
                        readBuffer[readBufferPosition++] = b;
                    }
                }
                if(AppConstant.MODE_CONNECT_HANDLER){
                    Message msg = mHandler.obtainMessage(ModeTagProtocol.MESSAGE_GPROOF);
                    Bundle bundle = new Bundle();
                    bundle.putString(ModeTagProtocol.GPROOF, data);
                    msg.setData(bundle);
                    mHandler.sendMessage(msg);
                }else{
```

Figura 46: Hilo para realizar transmisiones de datos.

- **DeviceListActivity.java:** esta clase nos proporciona la funcionalidad para buscar y emparejar dispositivos.

```
    // Request discover from BluetoothAdapter
    mBtAdapter.startDiscovery();
}
```

Figura 47: Buscar dispositivos bluetooth.

```
private void pairDevice(BluetoothDevice device) {
    try {
        if (D) Log.d(TAG, "Start Pairing...");
        Method m = device.getClass().getMethod("createBond", (Class[]) null);
        m.invoke(device, (Object[]) null);
    } catch (Exception e) {
        Log.e(TAG, e.getMessage());
    }
}
```

Figura 48: Emparejar dispositivos bluetooth.

Paquete "crypt"

Este paquete se encarga de cifrar y descifrar datos mediante la clase *SimpleCrypto.java*, la cual cifra con el método AES 128 bits proporcionado por la API de java.

Paquete "yoking-proof"

Este paquete nos proporciona las opciones de la aplicación mediante tres clases:

- **SplashScreen.java:** proporciona el inicio de aplicación haciendo uso de un hilo.

```
new Handler().postDelayed(new Runnable() {

    /*
     * Showing splash screen with a timer.
     */

    @Override
    public void run() {
        // This method will be executed once the timer is over
        // Start your app main activity
        Intent i = new Intent(SplashScreen.this, GProof.class);
        startActivity(i);

        // close this activity
        finish();
    }
}, SPLASH_TIME_OUT);
```

Figura 49: Hilo que ejecuta el "Splash Screen" de la aplicación.

- **GProof.java:** proporciona la ventana principal donde podemos pasar a la configuración de los dos modos o acceder al menú para emparejar dispositivos, hacerlo visible, administrar TAGs/Grupos y Acerca de. Además, como es la que se muestra al inicio tras la ventana "*SplashScreen*", nos pregunta si activar el bluetooth o no:

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
} else { // Otherwise, setup the UI
```

Figura 50: "Intent" para activar el bluetooth en cada una de las ventanas.

- **ManageTagGroup.java:** proporciona una interfaz con dos pestañas *(fragments)*, las cuales permiten añadir o consultar TAGs/Grupos de la base de datos.

Paquete "tab_manage_adapter"

En este paquete nos encontramos con tres clases **GroupFragment.java, TabPagerAdapter.java y TagFragment.java**, las cuales nos proporcionan lo comentado en el paquete anterior. La clase TabPagerAdapter es la encargada de darnos el soporte para el uso de las pestañas, así como su gesto "*slide*".

Paquete "yokingproof.helper"

En este paquete nos encontramos con una única clase java llamada **AppConstant.java** la cual nos proporciona la mayor parte de las variables globales para su modificación rápida y eficaz.

Paquete "yokingproof.reader_verifier"

En este paquete nos encontramos con dos clases:

- **ModeReader.java:** esta clase nos proporciona un formulario para configurar el dispositivo en modo Lector/Verificador. El formulario hace uso de *Textview*, *EditText*, *CheckBox*, *Spinners* (muestran TAGs/Grupos de la base de datos), y un *Button* para configurar las variables del protocolo.

- **ModeReaderProtocol.java:** esta clase nos proporciona un resumen en la interfaz de los datos configurados para el protocolo, así como de su estado de ejecución, y a su vez permite ejecutar o parar el mismo mediante una serie de botones. Contiene las funciones f1,f3,f5 y f7 del protocolo.

La implementación de la ejecución del protocolo se ha llevado a cabo con el uso de un Hilo (Thread) y de un Manejador (Handler) para poder visualizar datos necesarios en la interfaz además de gestionar la información recibida de los hilos creados por la clase *BTService.java* para el establecimiento de conexiones y posterior intercambio de información entre dispositivos.

Cuando se produce la perdida de conexión de algún dispositivo o intento de violación del protocolo por parte de algún atacante, éste nos hace percatarnos de ello avisándonos mediante un pitido y vibración, además de un dialogo indicándonos de cuál es el dispositivo (TAG) que está fallando.

Paquete "yokingproof.statistics"

Este paquete está compuesto por una sola clase, **StatisticsProtocol.java**, la cual nos proporciona un fichero en la memoria externa del dispositivo con los tiempos de cada pasada (desde f1() a f7()) del protocolo. Cada ejecución nos da los tiempos en milisegundos de los GAPs representados en la Figura 23 para su posterior estudio.

Paquete "yokingproof.tag"

En este paquete nos encontramos con dos clases:

- **ModeTag.java:** esta clase nos proporciona un formulario para configurar el dispositivo en modo TAG. El formulario hace uso de *Textview y Spinners* (muestran TAGs/Grupos de la base de datos), además de un *Button* para configurar las variables del protocolo.

- **ModeTagProtocol.java:** esta clase nos proporciona un resumen en la interfaz de los datos configurados para el protocolo, así como de su estado de ejecución, y a su vez permite ejecutar o parar el mismo mediante una serie de botones. Contiene las funciones f2, f4 y f6 del protocolo además de los PRNGs (XORshift y JKISS32).

```
// XORshift
static long XORshift(long max) {  // 2 7 7 ; 5 17 13
    seed ^= (seed << 5);    seed = seed % max;
    seed ^= (seed >> 17);   seed = seed % max;
    seed ^= (seed << 13);   seed = seed % max;
    long out = seed % max;
    return (out < 0) ? -out : out;
}
// JKISS32
static long JKISS32(){
    long t;
    y ^= y << 5;            y = y % 4294967296L;
    y ^= y >> 7;            y = y % 4294967296L;
    y ^= y << 22;           y = y % 4294967296L;
    t = z + w + c;          t = t % 4294967296L;
    z = w;                  z = z % 4294967296L;
    c = t << 0;             c = c % 4294967296L;
    w = t & 2147483647;     w = w % 4294967296L;
    x += 1411392427;        x = x % 4294967296L;
    return (x + y + w) % 4294967296L;
}

static void setSeed_JKISS32(long newseed )
{
    if ( newseed != 0 )
    {
        x = 123456789;
        y = newseed;
        z = 345678912;
        w = 456789123;
        c = 0;
    }
}

static void setSeed(long s) {
    seed = s;
}
```

Figura 51: Implementación PRNGs en Java.

La implementación de la ejecución del protocolo se ha llevado a cabo con el uso de un Hilo (*Thread*) y de un Manejador (*Handler*) para poder visualizar datos necesarios en la interfaz además de gestionar la información recibida de los hilos creados por la clase *BTService.java* para el establecimiento de conexiones y posterior intercambio de información entre dispositivos.

Cuando se produce algún intento de violación del protocolo por parte de algún atacante, éste nos hace percatarnos de ello avisándonos mediante un pitido y vibración, además de un dialogo indicándonos de qué está pasando.

Paquete "sqlite.helper"

Este paquete está dotado de la clase que gestiona toda la base de datos (Creación, Actualización, Inserción y Borrado) llamada **DatabaseHelper.java**.

```
@Override
public void onCreate(SQLiteDatabase db) {
    // *** BBDD Reader ***
    db.execSQL(CREATE_TABLE_TAG);
    db.execSQL(CREATE_TABLE_GROUP);
    db.execSQL(CREATE_TABLE_TAGGROUP);
    // *** BBDD Verifier ***
    db.execSQL(CREATE_TABLE_TIMESTAMP);
    db.execSQL(CREATE_TABLE_EVIDENCE);

    // Insert default data in Tag & Group
    String insertTag1 = "INSERT INTO " + TABLE_TAG + "( " + KEY_NAME + "," + KEY_ID_TAG + '
    String insertTag2 = "INSERT INTO " + TABLE_TAG + "( " + KEY_NAME + "," + KEY_ID_TAG + '
    String insertTag3 = "INSERT INTO " + TABLE_TAG + "( " + KEY_NAME + "," + KEY_ID_TAG + '
    String group1 = "INSERT INTO " + TABLE_GROUP +   "( " + KEY_NAME + "," + KEY_ID_GROUP+
    String group2 = "INSERT INTO " + TABLE_GROUP +   "( " + KEY_NAME + "," + KEY_ID_GROUP+

    db.execSQL(insertTag1);
    db.execSQL(insertTag2);
    db.execSQL(insertTag3);
    db.execSQL(group1);
    db.execSQL(group2);
}
```

Figura 52: Método onCreate() de la base de datos.

Paquete "sqlite.model"

Este paquete contiene una clase por cada tipo de tabla existente en la base de datos para tratar cada tipo de dato como un objeto y así facilitar la implementación. Estas clases son:

- **Evidence.java:** contiene el objeto *Evidence* con sus respectivos *constructors, getters and setters.*
- **Group.java:** contiene el objeto *Group* con sus respectivos *constructors, getters and setters.*
- **Tag.java:** contiene el objeto *TAG* con sus respectivos *constructors, getters and setters.*

- **TagGroup.java:** contiene el objeto *TagGroup* con sus respectivos *constructors, getters and setters.*
- **Timestamp.java:** contiene el objeto *Timestamp* con sus respectivos *constructors, getters and setters.*

Conclusión

Donde más problemas se han encontrado y donde más tiempo se ha dedicado es a la gestión de los Hilos (Threads) y Manejadores (Handlers), que gestionan la ejecución del protocolo, por su gran complejidad de uso.

Un aspecto importante a comentar es que no se han cifrado los identificadores ni las claves de TAG/Grupo porque Arduino no dispone de la capacidad de cifrado ni descifrado usada. Lo que si se ha cifrado es el sello de tiempo (*timestamp)*, pero aun así se ha tenido que hacer uso del formato de Java *BigInteger* para hacer una conversión a formato decimal y módulo del tamaño máximo permitido por Arduino para operar (unsigned long → desde 0 a 2^{32}).

5.2 PROGRAMA ADUINO – YOKINGPROOF.INO

La implementación del programa *"YokingProof.ino"* sobre la plataforma Arduino UNO se ha llevado a cabo mediante el lenguaje de programación C. El IDE empleado es la versión 1.5.7 Beta que proporcionan ellos mismos de una manera totalmente abierta y gratuita.

Tal y como se explicó en el Diseño, se ha hecho una división virtual para poder explicar de una forma más clara este fichero de la plataforma Arduino que ejecuta la parte de TAG del protocolo. No se ha implementado la parte de Lector/Verificador por disponer de una baja capacidad de procesamiento y memoria. Esta división de la que hablamos son las librerías, las constantes, las variables globales y los métodos.

Librerias

Se han introducido las siguientes librerías para dar soporte a funciones utilizadas en el desarrollo:

- **#include <SoftwareSerial.h>:** es una librería para usar otros pines como Puerto serial para el dispositivo bluetooth, dejando así libre el puerto serial por defecto de Arduino (pines 0 y 1) para *debug*. it is to use another serialport and can use default Serial to debug.

- **#include <string.h>:** proporciona funciones para el tratamiento de Strings.
- **#include <.h>:** es la librería estándar que proporciona funciones básicas, como la conversión de cadena de caracteres a enteros de tamaño largo *(atol)* usada en la implementación.

Variables y constantes

Se han usado con la finalidad de dar cierta usabilidad al código para cambiar valores rápidamente, como por ejemplo pueden ser los identificadores de TAG y de Grupo, además de las claves.

Métodos

Aquí hay dos funciones principales que todo código de Arduino debe tener, éstas son:

- **setup():** donde hacemos todas la incializaciones y configuraciones, como por ejemplo para el debug, el LED, el altavoz, el dispositivo bluetooth mediante comandos AT…

```
BTSerial.print("AT"); delay(DELAY_CONFIG_BT);// This delay is required. Delay re
// Set BT Name
BTSerial.print("AT+NAME"); BTSerial.print(ssid); delay(DELAY_CONFIG_BT);
// Set BaudRate
BTSerial.print("AT+BAUD"); BTSerial.print(baud_rate); delay(DELAY_CONFIG_BT);
// Set Pass:
BTSerial.print("AT+PIN"); BTSerial.print(pass); delay(DELAY_CONFIG_BT);
// Get Version:
//bluetooth.print("AT+VERSION"); delay(1500);
BTSerial.flush(); // Clean serial port memory
```

Figura 53: Comandos AT para configurar el dispositivo bluetooth HC-06.

- **loop():** esta función se ejecuta indefinidamente, y en ella lo que hacemos es escuchar cuándo la conexión bluetooth tiene datos que proporcionarnos, procesar los mismos y ejecutar la parte del protocolo correspondiente a dichos datos.

```
/* ******************** LOOP ******************** */
void loop()
{
    delay(5);

    // *** Start waitingData() ***
    Serial.println("Waiting data..."); // Debug
    //check if there's any data sent from the remote bluetooth shield
    if (BTSerial.available() < 1) {
        return; // if BTSerial empty, return to loop().
    }
    // *** END waitingData() ***

    groupingProof();
}
```

Figura 54: Funcion Loop() de la plataforma Arduino.

Contiene las funciones correspondientes con la parte TAG del protocolo f2(), f4() y f6(), además de los PRNGs (XORshift y JKISS32).

Conclusiones

Lo más complicado de esta plataforma es la complejidad para depurar los errores que han ido surgiendo. Esto se hace muy complicado ya que es una plataforma para implementaciones de sistemas en tiempo real, y la forma para ello es imprimir cada cosa que quieras explorar.

Otra cosa a comentar ha sido el ajuste en las variables que hemos tenido que hacer para que las plataformas Android (Java) y Arduino (C) se entiendan bien y sean capaces de ejecutar el protocolo correctamente.

CAPÍTULO 6

PRUEBAS

Después de haber realizado el Análisis, Diseño e Implementación del Sistema, en este capítulo vamos a comentar las pruebas más relevantes del mismo para verificar que cumple con los requisitos y mostrar posibles usos que pueden realizarse gracias a la implementación del protocolo propuesta.

6.1 DEFINICIÓN DEL PLAN DE PRUEBAS

Se realizarán una serie de pruebas de **aceptación** para satisfacer que el sistema cumple con el funcionamiento esperado.

Para definir más en detalle cada prueba, se va a usar la siguiente tabla modelo:

ID	PRU-Y-XX
Nombre	
Descripción	

Tabla 100: Tabla modelo para las pruebas de aceptación.

Descripción de cada campo de la Tabla 100:

- **ID:** código identificativo único con el formato PRU-Y-XX, siendo Y el identificador para saber si se trata de una prueba de la aplicación Android (A) o del programa Arduino UNO (AU). Las XX representan el número de prueba iniciado en 00.
- Nombre
- **Descripción:** explicación de las pruebas llevadas a cabo para verificar el cumplimiento del requisito.

Catálogo de pruebas aplicación Android

ID	PRU-A-01
Nombre	Bluetooh.
Descripción	Se comprobará que cada ventana de la aplicación te pide activar el bluetooth y si no se hace se sale automáticamente de la aplicación.

Tabla 101: PRU-A-01; Bluetooth.

ID	PRU-A-02
Nombre	Contraseña de cifrado.
Descripción	Se comprobará que al iniciar la aplicación se pide introducir una contraseña para cifrar contenido, y si no se hace se sale de la aplicación automáticamente.

Tabla 102: PRU-A-02; Contraseña de cifrado.

ID	PRU-A-03
Nombre	Emparejamiento, búsqueda y visibilidad Bluetooth.
Descripción	Se comprobará que la aplicación permite buscar y emparejar nuevos dispositivos, además de hacerlo visible.

Tabla 103: PRU-A-03; Emparejamiento y búsqueda Bluetooth.

ID	PRU-A-04
Nombre	Administración de TAGs/Grupos.
Descripción	Se comprobará que la aplicación permite administrar (añadir y consultar) los TAGs/Grupos que haya actualmente en la base de datos.

Tabla 104: PRU-A-04; Administración de TAGs/Grupos.

ID	PRU-A-05
Nombre	Consultar "Acerca de".
Descripción	Se comprobará que se puede acceder a la información de la aplicación además de poder enviar un correo al desarrollador.

Tabla 105: PRU-A-05; Consultar "Acerca de".

ID	PRU-A-06
Nombre	Configuración y Ejecución modo Reader/Verifier.
Descripción	Se comprobará que se puede configurar el formulario del modo Reader/Verifier para su posterior ejecución. El formulario deberá informar de los errores posibles que pueda haber, y la ejecución del protocolo deberá de avisar en caso de problema. Ésta además se podrá empezar y parar en cualquier momento. Además deberá poder ejecutarse en segundo plano.

Tabla 106: PRU-A-06; Configuración y Ejecución modo Reader/Verifier.

ID	PRU-A-07
Nombre	Configuración y Ejecución modo TAG.
Descripción	Se comprobará que se puede configurar el formulario del modo TAG para su posterior ejecución. La ejecución del protocolo deberá de avisar en caso de problema. Ésta además se podrá empezar y parar en cualquier momento. Además deberá poder ejecutarse en segundo plano.

Tabla 107: PRU-A-07; Configuración y Ejecución modo TAG.

Catálogo de pruebas aplicación Arduino UNO

ID	PRU-AU-01
Nombre	Bluetooh.
Descripción	Se comprobará que se puede realizar una nueva configuración del bluetooth cada vez que se quiera con un nuevo Nombre y PIN de conexión.

Tabla 108: PRU-AU-01; Bluetooth.

ID	PRU-AU-02
Nombre	Administración TAGs/Grupos.
Descripción	Se comprobará que se pueden cambiar los identificadores y claves de los TAGs/Grupos sin ningún problema.

Tabla 109: PRU-AU-02; Administración TAGs/Grupos.

ID	PRU-AU-03
Nombre	Ejecución del modo TAG.
Descripción	Se comprobará que se ejecuta el modo TAG sin ningún problema, y en el caso de que este ocurra, el dispositivo avisa con una señal luminosa y sonora.

Tabla 110: PRU-AU-03; Ejecución del modo TAG.

6.2 RESULTADO DEL PLAN DE PRUEBAS

El resultado obtenido de las pruebas realizadas en el apartado anterior se muestra en la siguiente tabla:

Identificador de la Prueba	Resultado
PRU-A-01	Superada
PRU-A-02	Superada
PRU-A-03	Superada
PRU-A-04	Superada
PRU-A-05	Superada
PRU-A-06	Superada
PRU-A-07	Superada
PRU-AU-01	Superada
PRU-AU-02	Superada
PRU-AU-03	Superada

Tabla 111: Resultado del plan de Pruebas.

El hecho de haber superado correctamente todas estas pruebas corrobora que la aplicación cumple con los requisitos de usuario establecidos, y que por lo tanto puede ser utilizada para realizar otras pruebas enfocadas a la investigación como las que se muestran en la siguiente sección.

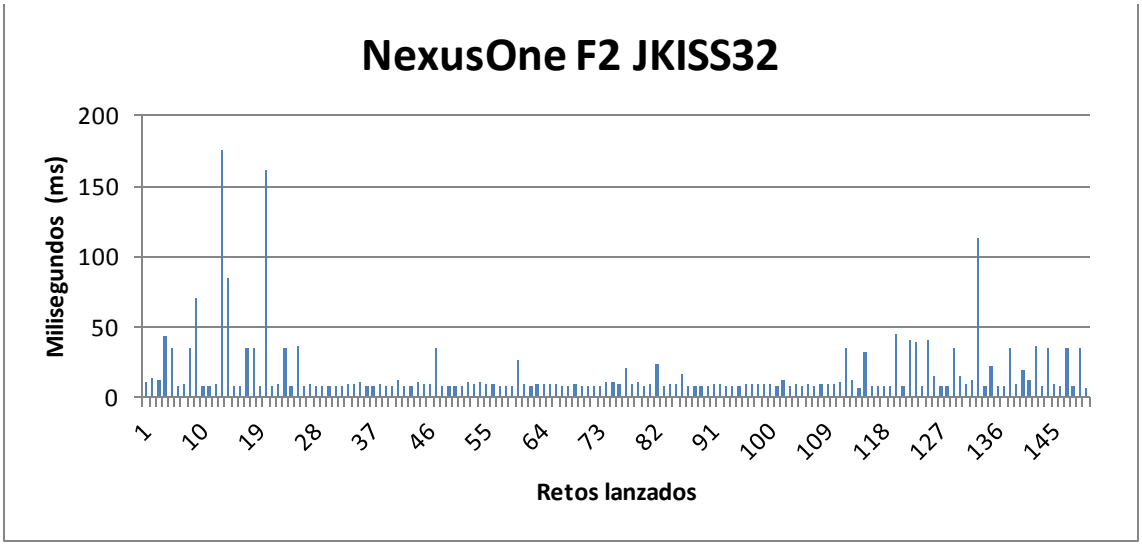
6.3 PRUEBAS DE INVESTIGACIÓN

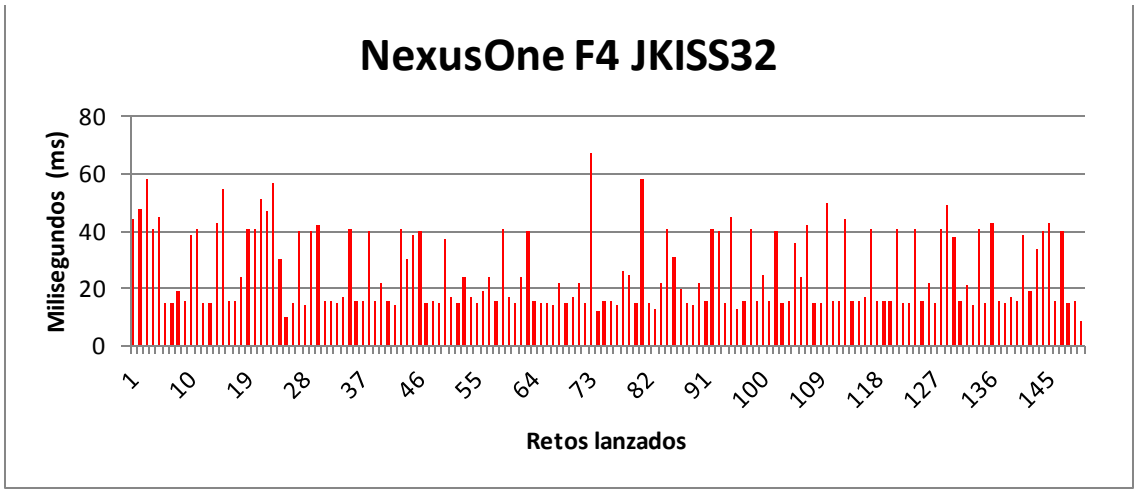
Para este apartado se han hecho una serie de pruebas especiales dedicadas más al ámbito de investigación.

Éstas se han realizado manteniendo el protocolo ejecutándose durante 5 minutos con el tiempo mínimo por prueba (2 segundos), a una distancia constante de unos 50 centímetros entre dos Smartphones (Google Nexus One y Google Nexus 5), ambos con la versión KitKat (v4.4.4), y una placa Arduino UNO. También se ha hecho cambiando el algoritmo de generación de números pseudoaleatorios (XORshift y JKISS32).

Con estas pruebas vamos a sacar los milisegundos que tarda en completarse cada función (véase Figura 23) en cada dispositivo, y usando cada función PRNG. Posteriormente, se una mostrarán los valores estadísticos de las pruebas (media, desviación típica, máximo y mínimo) y se realizará un análisis de los resultados obtenidos.

Graficas de resultados





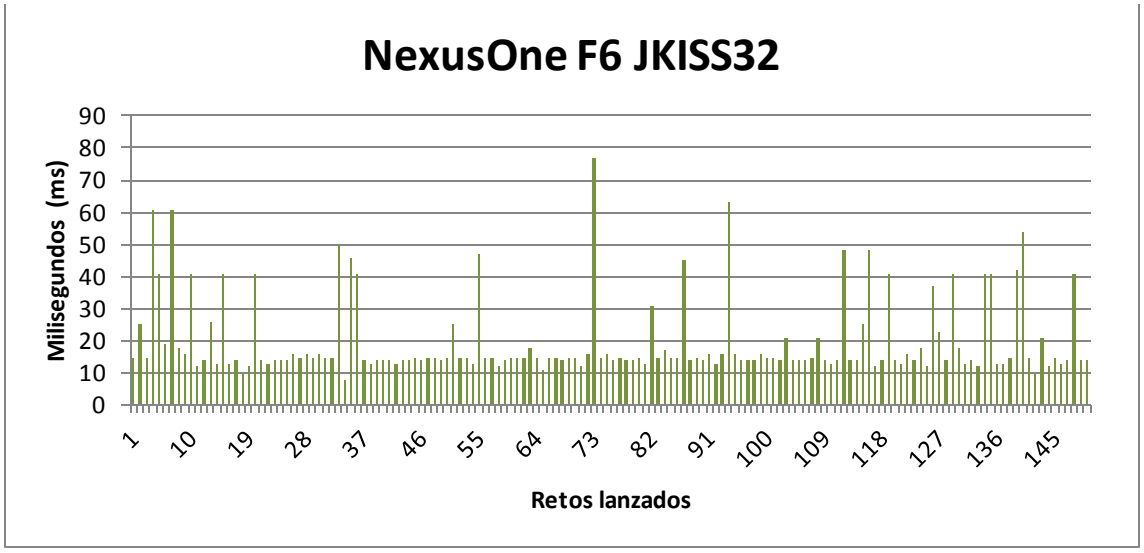
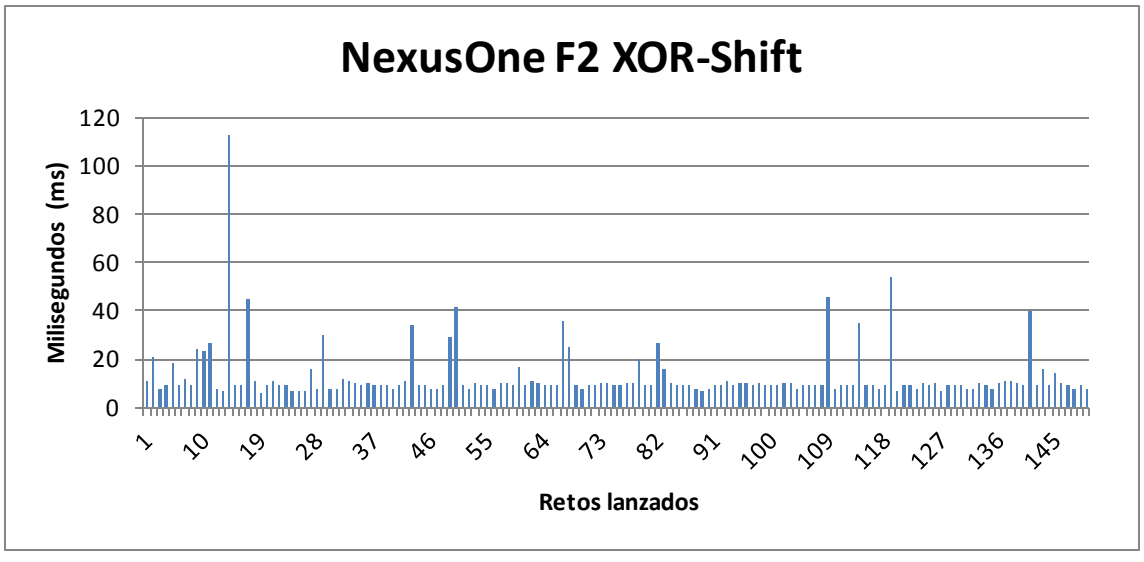
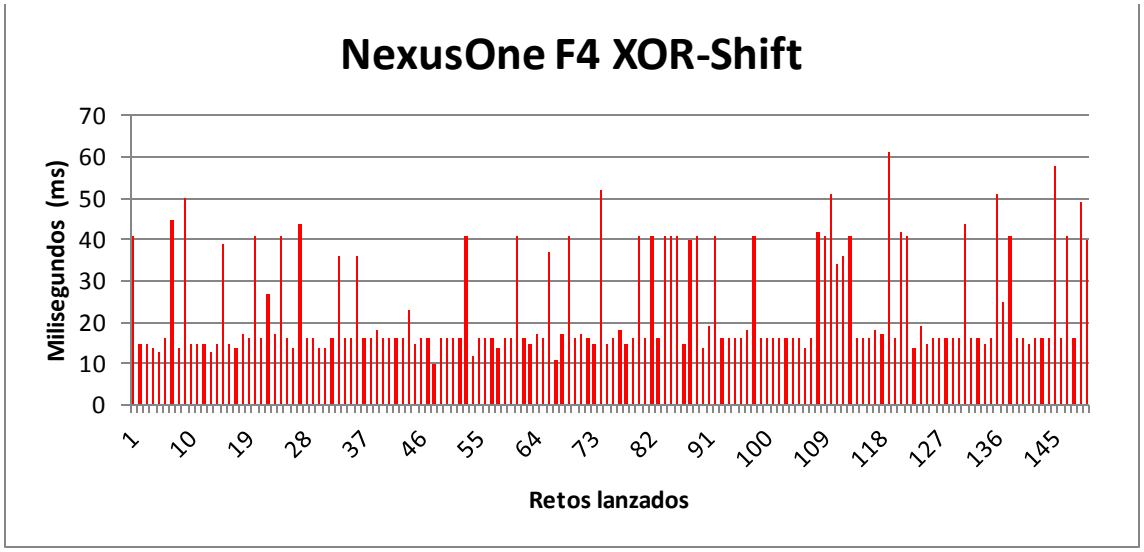


Figura 55: Tiempos de las funciones en NexusOne usando JKISS32.





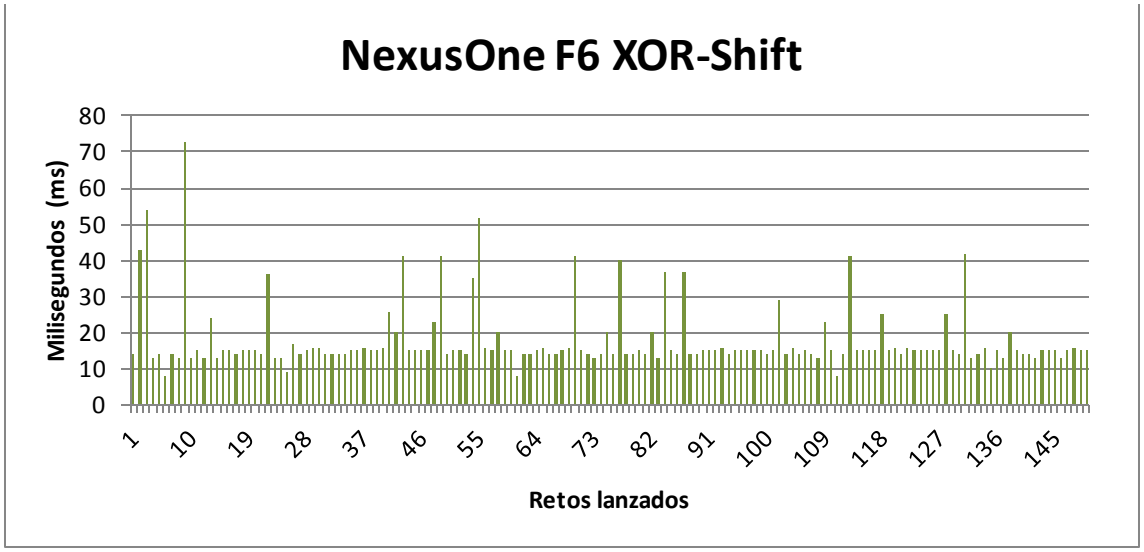


Figura 56: Tiempos de las funciones en NexusOne usando XORshift.

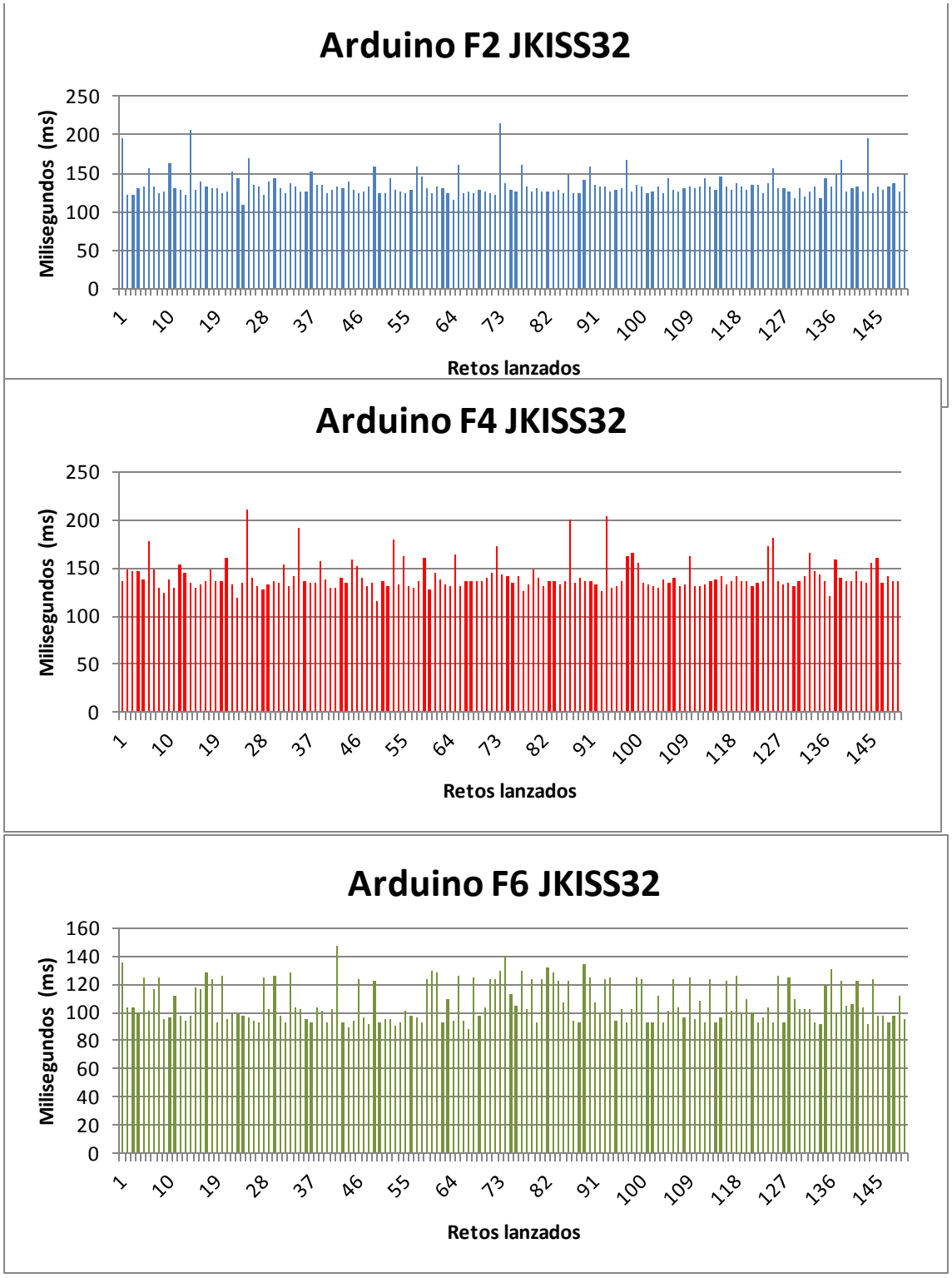


Figura 57:Tiempos de las funciones en Arduino usando JKISS32.

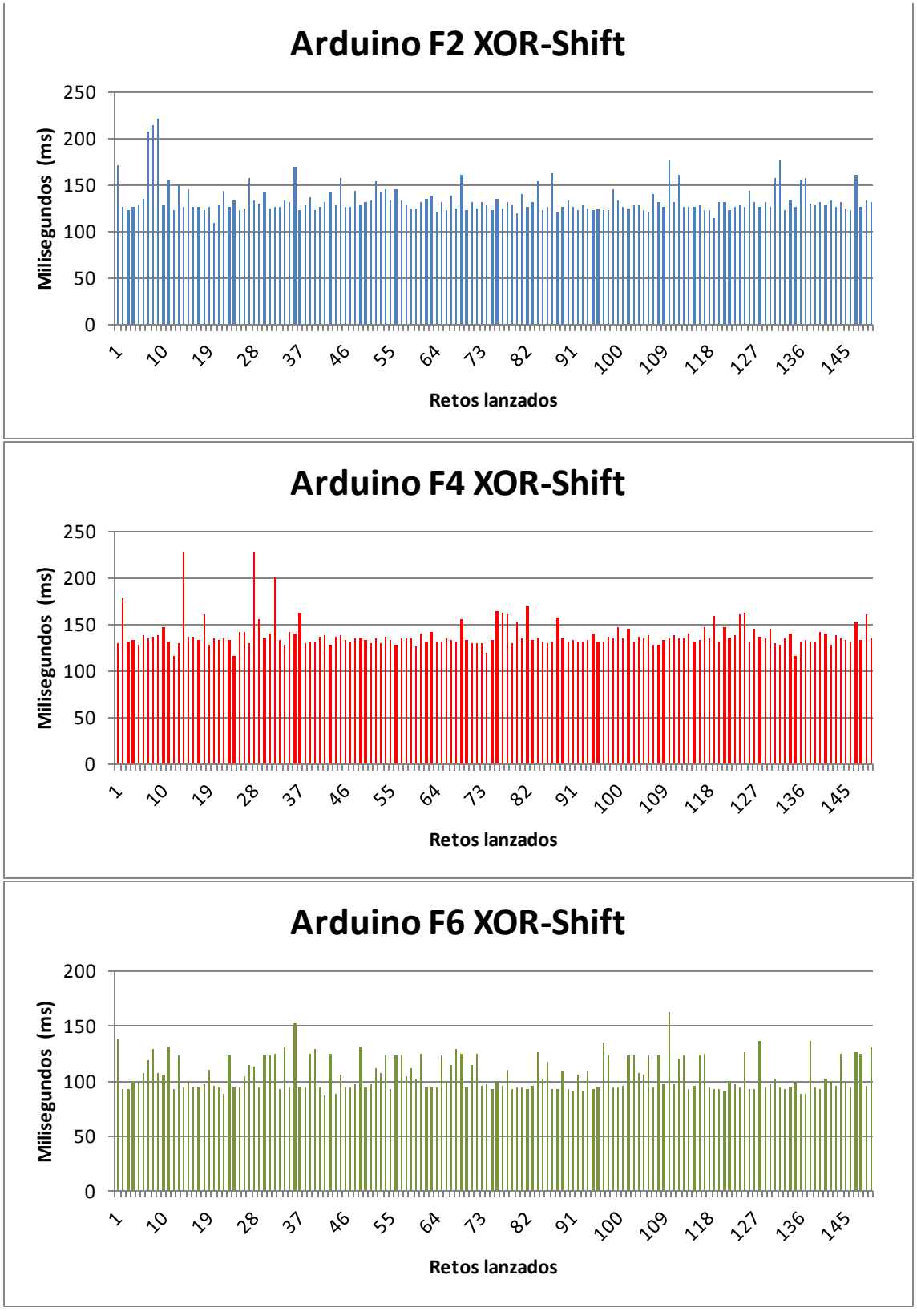


Figura 58: Tiempos de las funciones en Arduino usando XORshift.

Análisis de resultados

A continuación se muestran los valores estadísticos de las pruebas realizadas, dividiendo los resultados de utilizar ambas funciones de generación de números pseudoaleatorios (JKISS32 y XORShift), y mostrando en cada tabla los datos obtenidos para ambos dispositivos (NEXUS ONE y ARDUINO):

Tiempo en Milisegundo de las funciones usando JKISS32						
	NEXUS ONE			ARDUINO		
	F2	F4	F6	F2	F6	F6
MEDIA	18,74	25,92	20,55	135,15	140,90	106,54
DESV	23,07	14,01	12,91	17,60	14,80	14,67
MAX	176	84	77	278	211	183
MIN	7	9	8	109	115	88

Tabla 112: Resultados usando JKISS32.

Tiempo en Milisegundo de las funciones usando XORShift						
	NEXUS ONE			ARDUINO		
	F2	F4	F6	F2	F4	F6
MEDIA	12,86	22,38	17,73	133,84	138,72	106,57
DESV	11,81	12,05	9,13	15,74	14,51	15,40
MAX	113	61	73	221	228	163
MIN	6	10	8	110	116	87

Tabla 113: Resultados usando XORshift.

Como se puede observar con estos resultados, en todos los casos tarda más el algoritmo JKISS32 que XORShift, resultado que cabía esperar ya que el primero es más costoso y tiene más complejidad que el segundo.

En cuanto a la diferencia entre los dispositivos, puede observarse como tarda mucho menos el procesamiento en el Smartphone NexusOne que en el dispositivo Arduino. Este resultado también era de esperar, ya que el hardware de procesamiento del teléfono es mucho más potente que el del Arduino UNO.

Sin embargo, una conclusión interesante que puede extraerse de los resultados de las tablas es que, siendo el tiempo de ejecución medio mucho menor la desviación típica es semejante en ambos dispositivos. Es decir, aunque el tiempo sea menor, hay más variación en los resultados de una ejecución del protocolo a otro (lo que se corrobora con los distintos "picos" que aparecen en las gráficas relacionadas con Android. Esto puede ocurrir por varios factores, como conflictos entre aplicaciones en ese momento en el sistema operativo Android, ocupación de recursos, etc. Sin embargo, en un entorno real de aplicación del protocolo, estos "picos" podrían deberse a que el dispositivo haya sido comprometido y esté realizando alguna operación adicional al protocolo.

Es por ello que, a pesar de que en este Trabajo de Fin de Grado se ha implementado la función TAG en Android, es beneficioso que el tiempo de

procesamiento en distintas ejecuciones del protocolo sean similares, ya que de esta manera puede el Reader saber que el TAG no está haciendo "nada raro" adicional al protocolo. En todo caso, profundizar en este análisis y futuras mejoras al mismo quedan fuera del alcance de este Trabajo, y únicamente se pretende visualizar la utilizad de implementar un protocolo en dispositivos físicos para tener una visión de lo que ocurre en entornos reales.

CAPÍTULO 7

GESTIÓN DEL PROYECTO

En este capítulo se presenta la planificación del proyecto y el presupuesto asignado al mismo.

7.1 PLANIFICACIÓN

En este apartado se va a exponer la duración de las distintas tareas que se han llevado a cabo durante este proyecto. Para ello se va a hacer uso de la herramienta *"GanttProject"*, el cual nos va a generar el conocido diagrama de Gantt presentando de una forma gráfica la duración de cada una de las actividades.

La realización de este proyecto se ha divido en tareas las cuales se muestras a continuación con su fecha de inicio y con su fecha fin. En la siguiente tabla se muestran estos datos, los cuales nos servirán para realizar el diagrama de Gantt:

	Duración	Fecha de Inicio	Fecha de Fin
Lista de Objetivos	4 días	10/07/2014	13/07/2014
Estudio de las tecnologías a usar	10 días	14/07/2014	23/07/2014
Análisis	8 días	24/07/2014	31/07/2014
Diseño	7 días	01/08/2014	07/08/2014
Implementación	24 días	07/08/2014	31/08/2014
Pruebas	7 días	01/09/2014	07/09/2014
Documentación	17 días	08/09/2014	24/09/2014

Tabla 114: Planificación de Tareas.

Diagrama de Gantt:

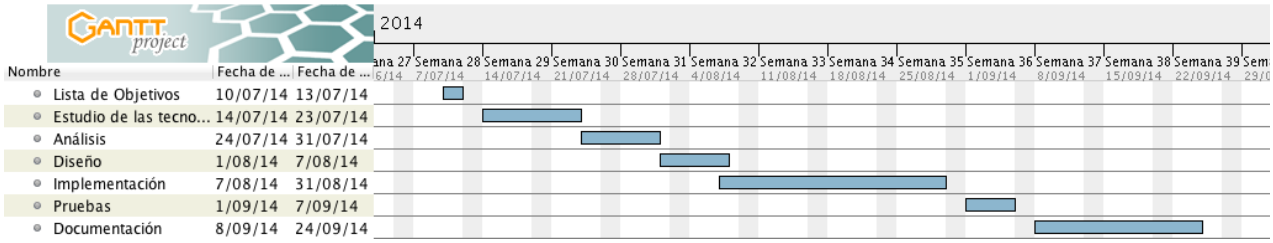


Figura 59: Diagrama de Gantt de la Planificación.

7.2 PRESUPUESTO

En este apartado se va a presentar el presupuesto para llevar a cabo la realización de este proyecto.

A continuación se van a desglosar los gastos:

- **Estimación de coste Personal:** se han dedicado 8 horas diarias. No ha habido días de vacaciones. Por lo tanto el numero de horas completas es 616 horas (8 horas x 77 días).

Concepto	Horas	Honorarios	Coste RRHH
Ingeniero Infomático	616 horas	20 €/hora	12320 €
PhD	30 horas	25 €/hora	750 €
TOTAL			13070 €

Tabla 115: Coste de personal Estimado.

- **Estimación de coste Hardware:** a continuación se presenta el hardware utilizado en el proyecto donde todos los precios tienen el IVA 21% incluido.

Concepto	Unidades	Precio Unitario	Vida útil estimada	Tiempo de uso	Coste para el proyecto
Apple MacBook Pro 13"	1	997,50 €	60 meses	2,56 meses	42,56 €
Google Nexus One	1	250 €	36 meses	2,56 meses	17,77 €
Google Nexus 5	1	400 €	36 meses	2,56 meses	28,44 €
Apple Magic Mouse	1	70 €	60 meses	2,56 meses	2,98 €
Arduino y Componentes	1	45 €	60 meses	2,56 meses	1,92 €
TOTAL					93,67 €

Tabla 116: Coste de Hardware Estimado.

- **Estimación de coste Software:** a continuación se presenta el software utilizado en el proyecto donde todos los precios tienen el IVA 21% incluido.

Concepto	Unidades	Precio Unitario	Vida útil estimada	Tiempo de uso	Coste para el proyecto
Apple Mac OS X 10.9.5	1	0 €	-	2,56 meses	0 €
Android Developer Tools	1	0 €	-	2,56 meses	0 €
Sublime Text 2	1	0 €	-	2,56 meses	0 €
Office 2013	1	119 €	36 meses	2,56 meses	8,46 €
LucidChart	1	0 €	-	2,56 meses	0 €
Dropbox 2.10.30	1	0 €	-	2,56 meses	0 €
GanttProject	1	0 €	-	2,56 meses	0 €
Arduino IDE 1.5.7 Beta	1	0 €	-	2,56 meses	0 €
SQLiteBrowser	1	0 €	-	2,56 meses	0 €
TOTAL					8,46 €

Tabla 117: Coste de Software Estimado.

- **Estimación de costes Indirectos:** para esta estimación de costes se han tenido en cuenta los costes de la Luz y del Internet.

Concepto	Precio Mensual	Tiempo de uso	Coste para el Proyecto
Luz	63 €	2,56 meses	161,28 €
Internet	35 €	2,56 meses	89,6 €
TOTAL			250,88 €

Tabla 118: Estimación de Costes Indirectos

- Estimación de costes Totales:

Concepto	Importe
Coste de Personal	13070 €
Coste de Hardware	93,67 €
Coste de Software	8,46 €
Costes Indirectos	250,88 €
TOTAL	13423,01 €

Tabla 119:Costes Totales del Proyecto.

Como se puede observar en la tabla 119, el coste total del proyecto incluyendo Personal, Software, Hardware e Indirectos, es de 13423,01 €.

CAPÍTULO 8

CONCLUSIONES

En este capítulo se exponen las conclusiones alcanzadas tras la finalización de este TFG, además de las líneas futuras en base a este desarrollo.

9.1 CONCLUSIONES GENERALES

El propósito principal de este TFG es analizar, adaptar e implementar un protocolo de seguridad para dispositivos móviles, en nuestro caso Android y Arduino, mediante bluetooth, comprobando que dos o más dispositivos pertenecen en el mismo momento a un mismo grupo.

El procedimiento para el desarrollo del protocolo, supone un gran trabajo a la hora de estudiar ambas plataformas, requiriendo que se entiendan entre sí, ya que cada una usa un lenguaje distinto (Java y C). También el estudio de la plataforma Bluetooth que juega un papel muy importante, puesto que es el medio por el cual comunicamos los dispositivos móviles y es accesible a cualquier atacante porque por el medio físico que funciona es el aire, y además el importantísimo el estudio de las operaciones de cifrado implicadas para la garantizar la seguridad.

Los resultados que hemos obtenido es que se garantiza la seguridad que nos provee el protocolo, además que puede ser un importante sistema para el futuro de los dispositivos móviles, puesto que tiene gran cantidad de aplicaciones en la vida real y que nosotros hemos conseguido simular.

Finalmente, decir que, lo más cotoso de este TFG ha sido el entendimiento entre plataformas, atendiendo también a la capacidad de procesado de cada una de ellas. El desarrollo de este TFG ha sido muy interesante, motivador y sobretodo gratificante, ya que he sido capaz de aplicar todos los conocimientos adquiridos durante el Grado, y por supuesto, he obtenido muchísimos nuevos.

9.2 LÍNEAS FUTURAS

En este apartado se van a presentar las posibles ampliaciones del sistema que se ha desarrollado.

Ampliar la experimentación

El sistema está desarrollado para un sistema de tres dispositivos móviles conectados por bluetooth, por lo que se podría extender a un número muy grande de dispositivos conectados al mismo tiempo, haciendo uso de una tecnología bluetooth nueva que no consuma tanta

batería. Esto puede requerir modificar el diseño actual seguramente, sobretodo en el apartado de gestión de conexiones.

A su vez, utilizando una técnica de bluetooth revolucionaria, se podría usar la distancia exacta entre dispositivos móviles para avisar en caso de que se salgan de un cierto rango (siendo menor éste que el rango bluetooth).

Integración de etiquetas RFID

Incluir el desarrollo etiquetas RFID revolucionarias, que posean una mayor capacidad de cómputo para poder ejecutar el protocolo con unas mayores condiciones de seguridad.

Implementación de otros protocolos de Yoking-Proofs

Incluir la implementación de protocolos con muchísima más capacidad de seguridad, aumentando la complejidad de los PRNGs, y que hagan un estudio aleatorio de los dispositivos que tengan en el mismo rango y grupo en ese mismo momento.

ANEXOS

ANEXO I. BIBLIOGRAFÍA

Bibliografía

(Intel), M. W. (18 de Junio de 2014). *ART vs Dalvik - Introducing the New Android* x86 Runtime*. Obtenido de https://software.intel.com/es-es/blogs/2014/06/18/art-vs-dalvik-introducing-the-new-android-x86-runtime

A CHILD'S GUIDE to BASIC TWO WAY BLUETOOTH COMMUNICATION. (s.f.). Obtenido de Scribd: www.scribd.com/doc/205888843/GUIDE-2BT

Arduino. (s.f.). *Arduino*. Obtenido de http://arduino.cc/

Arduino Tiny. (15 de Junio de 2012). *Arduino Tiny - Arduino programming for ATtiny processors*. Obtenido de Random: Overview: http://zygomorphic.com/arduino-tiny/?tag=random

Cabezudo, V. (1 de Marzo de 2014). *La historia de Internet of Things*. Obtenido de http://www.muycanal.com/2014/03/01/historia-internet-of-things

Cisco. (2014). *Internet of Things*. Obtenido de http://www.cisco.com/web/solutions/trends/iot/overview.html

Columbus, L. (17 de Enero de 2013). *2013 Roundup of Smartphone and Tablet Forecasts & Market Estimates*. Obtenido de http://www.forbes.com/sites/louiscolumbus/2013/01/17/2013-roundup-of-mobility-forecasts-and-market-estimates/

Evans, D. (Abril de 2011). *Cisco - Internet of Things.* Obtenido de La próxima evolución de Internet lo está cambiando todo: http://www.cisco.com/web/ES/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf

Google. (s.f.). *Android - Gingerbread*. Obtenido de https://developer.android.com/about/versions/android-2.3-highlights.html

Google. (s.f.). *Android - Honeycomb*. Obtenido de https://developer.android.com/about/versions/android-3.0-highlights.html

Google. (s.f.). *Android - Ice Cream Sandwich*. Obtenido de https://developer.android.com/about/versions/android-4.0-highlights.html

Google. (s.f.). *Android - Jelly Bean*. Obtenido de https://developer.android.com/about/versions/jelly-bean.html

Google. (s.f.). *Android KitKat*. Obtenido de https://developer.android.com/about/versions/kitkat.html

Google. (s.f.). *Android NDK*. Obtenido de https://developer.android.com/tools/sdk/ndk/index.html

Google. (s.f.). *Android SDK*. Obtenido de https://developer.android.com/sdk/index.html

Guangzhou HC Information Technology Co., L. (2011). Product Data Sheet HC-06. En L. Xin.

IDC. (2014). *Smartphone OS Market Share, Q2 2014*. Obtenido de http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Jaquero, C. A. (17 de Marzo de 2014). *Internet of things (IoT). El lado "Inseguro" de las Cosas*. Obtenido de http://www.inteco.es/blogs/post/Empresas/BlogSeguridad/Articulo_y_comentarios/Internet_of_things_ciberseguridad

Junichiro SAITO, K. S. (2005). Grouping proof for RFID tags. *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*.

La historia de Android. (2014). Obtenido de http://www.descargarandroid.com/la-historia-de-android/

Pedro Peris-Lopez, A. O.-C. (2011). Flaws on RFID grouping-proofs. Guidelines for future sound protocols. *Journal of Network and Computer Applications*(34), 833–845. Obtenido de www.elsevier.com/locate/jnca

Toro, L. M. (s.f.). *SISTEMAS DE IDENTIFICACIÓN POR RADIOFRECUENCIA*. Obtenido de http://www.it.uc3m.es/jmb/RFID/rfid.pdf

UC3M. (s.f.). *Software de Comunicaciones*. Obtenido de Arquitectura Android: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

Wikipedia. (2014). *RFID (Radio Frequency IDentification)*. Obtenido de https://en.wikipedia.org/wiki/Radio-frequency_identification

Wikipedia. (s.f.). *Android*. Obtenido de https://es.wikipedia.org/wiki/Android

Wikipedia. (s.f.). *List of random number generators*. Obtenido de https://en.wikipedia.org/wiki/List_of_random_number_generators

Wikipedia. (s.f.). *Mersenne twister*. Obtenido de https://en.wikipedia.org/wiki/Mersenne_twister

Wikipedia. (s.f.). *Pseudorandom number generator*. Obtenido de https://en.wikipedia.org/wiki/Pseudorandom_number_generator

ANEXO II. MANUAL DE APLICACIONES

En este anexo se presentan los manuales de usuario para el correcto uso de las aplicaciones desarrolladas en este proyecto.

Manual "Grouping Proof" (Aplicación Android)

Lo primero de todo es tener la aplicación instalada mediante el APK correspondiente:

Figura 60: Aplicación "Grouping Proof".

Al iniciarla nos encontraremos con una ventana de bienvenida y la ventana principal de la aplicación, después de haber activado el bluetooth y de haber introducido la contraseña de cifrado:

Figura 61: Ventana principal "Grouping Proof"

En esta ventana, como se puede observar, consta de un menú superior y de los botones de los dos modos de ejecución del protocolo. El menú situado en la barra de arriba nos permite Buscar y Emparejar dispositivos, Administrar los TAGs/Grupos, hacer el dispositivo Visible y consultar la información de la aplicación Acerca de.

Si elegimos la ejecución del modo Reader/Verifier, tenemos que completar un formulario seleccionado dos dispositivos móviles diferentes y distintos marcando si se trata de dispositivos Android o Seriales, los IDs que les queramos asignar distintos también, un grupo y el tiempo por Reto que se quiere emplear.

Figura 62: Configuración Modo Reader/Verifier.

Si elegimos la ejecución del modo TAG, hay que elegir el ID y el Grupo que le hemos asignado en el modo Reader/Verifier.

Figura 63: Configuración Modo TAG.

Finalmente, después de la configuración de cada uno de los modos, nos encontramos con las ventanas de ejecución de los mismos. Los dispositivos que tienen que empezar el protocolo son los TAGs ya que tienen que esperar a ser identificados mediante Bluetooth. Posteriormente se ejecuta el modo Reader/Verifier.

Figura 64: Ejecución Modo Reader/Verifier.

Figura 65: Ejecución Modo TAG.

Manual "Grouping Proof" (Programa Arduino)

En este programa implementado en Arduino, lo único que se nos permite es la ejecución del Modo TAG y la configuración del Bluetooth.

Primero, lo que hay que hacer, es configurar las variables Bluetooth en el caso de que queramos modificar los datos de conexión. Estas variables son CONFIG_BT, ssid (nombre) y pass (pin):

```
// *** Bluetooth Variables ***
#define CONFIG_BT 0  // 0 -> no configure bluetooth ; 1 -> configure bluetooh
char ssid[10] = "TAG1";     // BT name device. Max 20 char
char pass[5] = "1234";      // KEY to pair device. Max 4 char
```

Figura 66: Configuración Bluetooth Arduino.

Tras esta configuración del bluetooth, lo que hacemos es configurar las variables del Modo TAG, es decir, los identificadores y claves TAG/Grupo:

```
// *** Grouping-Proof Protocol Variables ***
// IDs & Keys
// IMPORTANT: Only numbers from 0 to 4294967295,
//            if it is more than this MaxNumber, string2long() gets another different number.
//            If string2long() receives number+letter, gets number. If it is letter+number, gets 0.
String ID_Group = "1234";
String K_Group = "4321";
String ID_Tag = "123456";
String K_Tag = "654321";
```

Figura 67: Configuración Modo TAG Arduino.

Finalmente lo que tenemos que hacer es cargar compilar y cargar el código en la placa Arduino UNO con el IDE oficial de Arduino.